Submodular Maximization and its Applications,
with Recent Advances in Streaming/Distributed Computing

Jiecao Chen

March 6, 2016

1 Introduction

Submodularity is a property of set functions with deep theoretical and practical consequences. Submodular functions occur in a variety of applications, including representative skyline selection [42], network structure learning [22], influence maximization [25], document summarization [32], image segmentation [7, 26] and many others.

There is a large body of research in submodular optimization, and our main focus of this survey is submodular maximization. In particular, we will cover the recent advances of maximizing submodular functions in distributed and streaming setting. Part of the reason we do not discuss submodular minimization is that, one can convert a minimization problem to a convex optimization problem through the Lovász' extension [34], and it can be therefore solved in polynomial time. On the other hand, most submodular maximization problems we will discuss are NP-Hard, and can only be solved efficiently via approximation algorithms.

Roadmap: This survey is organized as follows. In Section 2 we introduce several important properties of submodular functions; we then discuss different set systems that serve as constraints in submodular optimization problems; we also cover several classical results for submodular maximization in RAM model. In Section 3 we discuss the applications of submodular maximization, including both classical problems and formulations appear in recent research. We then cover recent progress made in the area of streaming/distributed submodular maximization in Section 4 and Section 5. A conclusion and possible further research are discussed in Section 6.

Notations: Through out this survey, we use V or E to represent the ground set we consider; 2^V is the power set of V (i.e. the set of all subsets of V); in general, A^V is the collection of maps from V to A; for simplicity, we sometime write $A \cup \{x\}$ as $A + x$. Other notation will be defined later when it is first introduced.

2 Submodularity

In this section, we first give several equivalent definitions of submodularity, and then we introduce several fundamental properties of submodular functions. We also discuss various constraints that occur frequently in submodular optimization problems. In the last part of this section, we

cover algorithms that solve constrained submodular maximization problems with theoretical approximation guarantee.

2.1 Definitions

There are many equivalent definitions, and we discuss three of them in this section.

Definition 1 (submodular concave) A function $f : 2^V \to \mathbb{R}$ is submodular if for any $A, B \subseteq V$, we have that:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B). \tag{1}$$

An alternate equivalent definition is more interpretable in many situations,

Definition 2 (diminishing returns) A function $f : 2^V \to \mathbb{R}$ is submodular if for any $A \subseteq B \subset V$, and $v \in V \backslash B$, we have that:

$$f(A + v) - f(A) \geq f(B + v) - f(B). \tag{2}$$

Intuitively, this definition requires that the incremental "gain" of adding a new element v decreases (diminishes) as the base set grows from A to B. We will see that this property is actually shared by many real-world phenomenons.

It turns out that a stronger but equivalent statement can also serve as the definition of a submodular function,

Definition 3 (group diminishing returns) A function $f : 2^V \to \mathbb{R}$ is submodular if for any $A \subseteq B \subset V$, and $C \subseteq V \backslash B$, we have that:

$$f(A \cup C) - f(A) \geq f(B \cup C) - f(B). \tag{3}$$

2.2 Modularity and Supermodularity

We also briefly mention modularity and supermodularity here. These two concepts are closely related to submodularity.

A function $f : 2^V \to \mathbb{R}$ is modular if we replace inequality by equality in Definition 2 (or any of other two). Formally,

Definition 4 (Modularity) A function $f : 2^V \to \mathbb{R}$ is modular if for any $A \subseteq B \subset V$, and $v \in V \backslash B$, we have that:

$$f(A + v) - f(A) = f(B + v) - f(B). \tag{4}$$

Notably, a modular function f can always be written as

$$f(S) = f(\emptyset) + \sum_{v \in S} (f(\{v\}) - f(\emptyset))$$

for any $S \subseteq V$. If we further assume $f(\emptyset) = 0$ (in this case, we call f normalized or proper), we have a simplified expression,

$$f(S) = \sum_{v \in S} f(\{v\}).$$

We can interpret a normalized modular function as a weight function: each element in the ground set is assigned a weight and given a set $S \subseteq V$, the modular function returns sum of weights in that set.

Modularity can be useful in our discussion of submodularity, because modular functions can be used to construct submodular functions with desired properties in many applications. Examples can be found in e.g. [32, 33].

A supermodular function is defined by flipping the inequality sign in the definition of a submodular function. Formally,

Definition 5 (Supermodularity) A function $f : 2^V \to \mathbb{R}$ is supermodular if for any $A \subseteq B \subset V$, and $v \in V \backslash B$, we have that:

$$f(A + v) - f(A) \leq f(B + v) - f(B). \tag{5}$$

We will focus on submodular functions because a function is supermodular if and only if its negative is submodular.

2.3 Properties

Like convex and concave functions, submodular functions have many nice properties. Lovász's comment on convex functions [34] gives an accurate description of submodular functions as well. We would like to quote it here,

> - Convex functions occur in many mathematical models in economy, engineering, and other sciences. Convexity is a very natural property of various functions and domains occurring in such models; quite often the only non-trivial property which can be stated in general.
>
> - Convexity is preserved under many natural operations and transformations, and thereby the effective range of results can be extended, elegant proof techniques can be developed as well as unforeseen applications of certain results can be given.
>
> - Convex functions and domains exhibit sufficient structure so that a mathematically beautiful and practically useful theory can be developed.
>
> - There are theoretically and practically (reasonably) efficient methods to find the minimum of a convex function.

We survey several useful properties which can be useful in our later sections. More properties of submodularity can be found in e.g. [5, 21].

Submodularity is closed under addition, formally,

Property 1 *Let $f_1, f_2 : 2^V \to \mathbb{R}$ be two submodular functions. Then*

$$f : 2^V \to \mathbb{R} \text{ with } f(A) = \alpha f_1(A) + \beta f_2(A)$$

is submodular for any fixed $\alpha, \beta \in \mathbb{R}^+$.

Adding a modular function does not break submodularity,

Property 2 *Let $f_1, f_2 : 2^V \to \mathbb{R}$, f_1 is submodular and f_2 is modular. Then*

$$f : 2^V \to \mathbb{R} \text{ with } f(A) = f_1(A) + \alpha f_2(A)$$

is submodular for any fixed $\alpha \in \mathbb{R}$.

Submodularity is preserved under restriction,

Property 3 *Let $f : 2^V \to \mathbb{R}$ be a submodular function. Let $S \subseteq V$ be a fixed set. Then*

$$f' : 2^V \to \mathbb{R} \text{ with } f'(A) = f(A \cap S)$$

is submodular.

As a direct implication of Property 1 and Property 3, we have the following more general result,

Property 4 *Let $f : 2^V \to \mathbb{R}$ be a submodular function, $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ be a collection of subsets of V (i.e. each $C_i \subseteq V$). Then*

$$f' : 2^V \to \mathbb{R} \text{ with } f'(A) = \sum_{C \in \mathcal{C}} f(A \cap C)$$

is submodular.

This property can be useful in graphical models and image processing. CHEN : *TODO: show examples*

The following property can be useful when we show that the objective function of k-medoid problem is supermodular,

Property 5 *Consider V as a set of indices. Let $\mathbf{c} \in \mathbb{R}^V$ be a fixed vector, c_i its ith coordinate. Then*

$$f : 2^V \to \mathbb{R} \text{ with } f(A) = \max_{j \in A} c_i$$

is submodular.

We can use non-negative modular function and a concave function to construct submodular functions,

Property 6 *Let $m : 2^V \to \mathbb{R}^+$ be a modular function, and f a concave function over $\mathbb{R}$. Then*

$$f : 2^V \to \mathbb{R} \text{ with } f(A) = g(m(A))$$

is submodular.

Before introducing the next property, we define the monotonitcity of set function,

Definition 6 (Monotonitcity) An set function $f : 2^V \to \mathbb{R}$ is said to be non-decreasing if for any $A \subseteq B \subseteq V$, $f(A) \leq f(B)$. Non-increasing set functions are defined in the similar way.

When we say a submodular function is monotone, we mean it is non-decreasing.

Property 7 *Let $f, g : 2^V \to \mathbb{R}$ be two submodular functions. If $(f - g)(\cdot)$ is either non-decreasing or non-increasing, then $f : 2^V \to \mathbb{R}$ with*

$$f(A) = \min(f(A), g(A))$$

is submodular.

2.4 Constraints

Now we discuss the constraints in submodular optimization problems. A submodular maximization problem usually has the following form,

$$\arg\max_{I \in \mathcal{I}} f(I) \tag{6}$$

where f is a submodular function and $\mathcal{I} \subseteq 2^V$ is the collection of all feasible solutions. We call $\mathcal{I}$ the constraint of the optimization problem. The structure of $\mathcal{I}$ plays a crucial role in submodular optimization. Different constraints have different hardness results, normally the difficulty increases when the constraint becomes more general. In this section, we define several of the most popular constraints. Before introducing any of them, we define

Definition 7 (Hereditary) A constraint $\mathcal{I} \subseteq 2^V$ is said to be hereditary if

$$I \in \mathcal{I} \implies J \in \mathcal{I} \text{ for any } J \subseteq I.$$

A hereditary constraint is sometimes called an independent system and each $I \in \mathcal{I}$ is called an independent set.

All constraints we discuss in this section are hereditary.

2.4.1 Cardinality Constraint

Cardinality constraint is perhaps the most straightforward yet most popular constraint we would discuss in this survey. Efficient algorithms have been developed for finding or approximating the optimal solution of (6) subject to this constraint. There are also a lot of extensions, to both streaming and distributed setting.

A cardinality constraint is parameterized with a fixed constant $k \in \mathbb{Z}^+$. It is simply defined as $\mathcal{I} = \{A \subseteq V \mid |V| \leq k\}$, i.e. all subsets of V with size no larger than k. Cardinality constraint is arguably the most popular constraint, and it occurs everywhere. For example, in k-medoid clustering, we want to find a set S of *at most* k points, that minimizes the total distance of all points to S.

2.4.2 Knapsack Constraint

Knapsack constraint generalizes cardinality constraint by assigning each element in V a weight. Given a budget $W > 0$ and assume that each $i \in V$ is assigned a weight $w_i \geq 0$, a knapsack constraint can be defined as $\mathcal{I} = \{S \subseteq V \mid \sum_{i \in S} w_i \leq W\}$.

2.4.3 Matroid

Informally, matroid is the abstraction of the *independence* concept in linear algebra. In fact there are so many results around matroid and the matroid itself becomes a subfield of algebra. We cover some basics of matroid theory, from which readers can easily see how powerful this concept is.

Before discussing the concept of matroid, we briefly review the independence concept from linear algebra. For simplicity, let us just consider $\mathbb{R}^d$ instead of a general linear space. A subset S of $\mathbb{R}^d$ is said to be *independent* if there does not exist any $\mathbf{x} \in S$ such that $\mathbf{x}$ can be represented by linear combination of vectors in $S \backslash \{\mathbf{x}\}$.

Let $\mathcal{I} = \{S \subseteq \mathbb{R}^d \mid S \text{ is independent }\}$, i.e. only a independent set can be considered feasible. From what we learn in college linear algebra course, we know $\mathcal{I}$ has the following propertie: 1) if $I \in \mathcal{I}$, any of I's subsets is also in $\mathcal{I}$ (or equivalently, $\mathcal{I}$ is hereditary); 2) if $J, I \in \mathcal{I}$ and J has smaller size than I, we must be able to find an element $\mathbf{x} \in I \backslash J$ such that $J \cup \{\mathbf{x}\} \in \mathcal{I}$.

Even for the "trivial" size function $f : 2^V \to \mathbb{Z}$ with $f(A) = |A|$, solving $\arg\max_{I \in \mathcal{I}} f(S)$ would have tremendous applications because its optimal solution is a base of the vector space. If we somehow generalize the definition of independence, we may be able to model a much more larger class of problems into the form (6).

It turns out that the properties of $\mathcal{I}$ we just described are sufficient to give a meaningful definition for Matroid. Formally,

Definition 8 (Matroid) A set system $(V, \mathcal{I})$ is a matroid if it has the following properties,

1. $\forall I \in \mathcal{I}, J \subseteq I \implies J \in \mathcal{I}$ (i.e. hereditary)
2. $\forall I, J \in \mathcal{I}$, with $|I| = |J| + 1$, then $\exists\, x \in I \backslash J$ such that $J \cup \{x\} \in \mathcal{I}$

Unlike in the $\mathbb{R}^d$ case, in this survey we restrict our discuss on matriod with a finite ground set. Note that in general, the intersection of several matroids is no longer a matroid.

Finally, we generalize the concept of rank in linear algebra. Let $(V, \mathcal{I})$ be a matroid, we define the rank function $r : 2^V \to \mathbb{Z}$ as $f(S) = \max_{I \subseteq S, I \in \mathcal{I}} |I|$, i.e. the rank of $S \subseteq V$ is the maximum possible size of S's subsets that are also members of $\mathcal{I}$ (or in other words, independent). Our definition of rank is consistent with what we have in linear algebra (in that case $\mathcal{I}$ is the collection of all independent sets). A rank function is submodular (as you may expect).

2.4.4 Matching

A matching of a graph $G = (V, E)$ is a set $S \subseteq E$ such that no edges in S share common vertex.

2.4.5 p-Matchoid

Let $\mathcal{M}_1 = (V_1, \mathcal{I}_1), \ldots, \mathcal{M}_q = (V_q, \mathcal{I}_q)$ be q matroids where $V = V_1 \cup \ldots V_q$. Let $\mathcal{I} = \{S \subseteq V \mid S \cap V_i \in \mathcal{I}_i \text{ for all } i\}$. The finite set system $(V, \mathcal{I})$ is a p-matchoid if for every $e \in V$, e is a member of V_i for at most p indices $i \in [q]$.

The concept of p-matchoid generalizes the intersection of matroids (taking $p = q$ and $V_i = V$ for all i), and matching (which is 2-matchoid).

constraint	monotone	non-negative
cardinality	$1-1/e$ [40]	$1/e+.004$ [8]
matroid	$1-1/e$ [10], R	$\frac{1-\epsilon}{e}$ [18], R
matching	$\frac{1}{2+\epsilon}$ [19]	$\frac{1}{4+\epsilon}$ [19]
intersection of p matroids	$\frac{1}{p+\epsilon}$ [30]	$\frac{p-1}{p^2+\epsilon}$ [30]
p-matchoid	$\frac{1}{p+1}$ [10, 40]	$\frac{(1-\epsilon)(2-o(1))}{e\cdot p}$ [19, 47], R

Table 1: Best known approximation bounds for submodular maximization in RAM model. Bounds for randomized algorithms that hold in expectation are marked (R).

2.4.6 p-System

p-system is the most general constraint we will discuss in this survey, it includes graph matching, p-matchoid (therefore matroid) and many others as special cases.

Let $(V, \mathcal{I})$ be a set system and $\mathcal{I}$ hereditary. An independent set I (i.e. $I \in \mathcal{I}$) is called a base of $A \subseteq V$ if $I \subseteq A$ and for any $e \in A \backslash I$, $I + e \notin \mathcal{I}$. Let $\mathcal{B}(A)$ be the collection of all bases of A, we call $(V, \mathcal{I})$ a p-system if it further satisfies the following property,

$$\forall A \subseteq V : \max_{S \in \mathcal{B}(A)} |S| \leq p \cdot \min_{S \in \mathcal{B}(A)} |S|.$$

It is known that p-matchoid is p-system (therefore matching is 2-system) and the intersection of p matroids $(V, \mathcal{I}_1), \ldots, (V, \mathcal{I}_p)$ is p-system.

2.5 Algorithms for Submodular Maximization

There are a lot of results for submodular maximization in the centralized setting where the data can fit into the main memory. Those results normally assume the oracle model: one is given a value oracle and a membership oracle. Given $S \subseteq V$, the membership oracle answers if $S \in \mathcal{I}$ and the value oracle returns $f(S)$. We cover several classical results which serve as the building blocks for distributed/streaming algorithms for submodular maximization. A summary of best known algorithms (in terms of approximation ratio) for various of constraints is presented in Table 1.

We introduce the notation for marginal gain: $\Delta_f(e|S) = f(S+e) - f(S)$, and we say an algorithm for sumbodular maximization gives α-approximation if the solution S it returns always has guarantee $f(S) \geq \alpha \cdot \arg\max_{I \in \mathcal{I}} f(I)$. When the algorithm is randomized, we normally say it guarantees α-approximation in expectation if $\mathbf{E}[f(S)] \geq \alpha \cdot \arg\max_{I \in \mathcal{I}} f(I)$.

The following algorithm shows a popular greedy strategy for submodular maximization.

Algorithm 1: GREEDY algorithm for submodular maximization

Input: V the ground set, f the submodular function, $\mathcal{I}$ the constraint
Output: a set $S \subseteq V$

```
S ← ∅
while ∃ e ∈ V\S s.t. S ∪ {e} ∈ I do
    e ← arg max_{e ∈ V\S, S∪{e} ∈ I} Δ_f(e|S)
    S ← S ∪ {e}
return S
```

2.5.1 Algorithms for Cardinality Constraint

A celebrated result of [40] shows that,

Theorem 1 ([40]) *For a non-negative monotone submodular function $f : 2^V \to \mathbb{R}$, let $\mathcal{I}$ be the cardinality constraint, Algorithm 1 returns a $(1 - 1/e)$-approximation to $\arg\max_{I \in \mathcal{I}} f(S)$.*

For several classes of submodular functions, this result is actually the best one can expect for any efficient algorithm (which we mean using only sub-exponential value queries). In fact the hardness result in [17, 40] shows that any algorithm that is only allowed sub-exponential number of value queries can not achieve better than $(1 - 1/e)$-approximation (for a large class of submodular functions).

There are several papers improving the running time of Algorithm 1 under cardinality constraint of size k (under which the membership oracle is trivial). Minoux [36] proposed LAZY-GREEDY as a fast implementation for Algorithm 1. Instead of computing $\Delta_f(e|S)$ for each $e \in V \backslash S$ in Line 3, LAZY-GREEDY keeps an upper bound $\rho(e)$ (initially $+\infty$) on the marginal gain sorted in decreasing order (or kept in a heap). In each iteration, the LAZY-GREEDY algorithm evaluates the element on top of the heap and updates its upper bound $\rho(e) \leftarrow \Delta(e|S)$. If the updated $\rho(e) \geq \rho(e')$ for all other e', submodularity guarantees that e is the element with the largest marginal gain. The exact number of value queries consumed by LAZY-GREEDY is unknown because it heavily relies on both f and V, experimental study however shows that the LAZY-GREEDY algorithm is in order of magnitude faster than the naive implementation of Algorithm 1.

Wei et al. [48] improved the running time of LAZY-GREEDY by approximating the underlying submodular function with a set of (sub)modular functions. Badanidiyuru et al. [3] proposed a different approach that uses only $O(\frac{|V|}{\epsilon} \log \frac{1}{\epsilon})$ number of value queries and guarantees $(1 - 1/e - \epsilon)$-approximation. This result was improved by Mirzasoleiman et al. [38] recently where they proposed a randomized algorithm (STOCHASTIC-GREEDY) that reduce the number of queries to $O(|V| \log \frac{1}{\epsilon})$ and the approximation guarantee is $(1 - 1/e - \epsilon)$, in expectation. The key observation made in [38] is that, instead of considering all $e \in V \backslash S$, one can only consider $O(\frac{|V|}{k} \log \frac{1}{\epsilon})$ random samples from $V \backslash S$.

2.5.2 Algorithms for Matroid and p-system

Now we consider the matroid constraint and p-system. In his classical paper [23], Jenkyns proved that for a non-negative *modular* maximization problem s.t. p-system, Algorithm 1 produces a p-approximation of the optimum. Since matroid is 1-system, Algorithm 1 always gives the optimal solution. Remarkably, we the following much stronger result,

Theorem 2 (see e.g. [5, 21]) *Let $(V, \mathcal{I})$ be a set system we consider, $\mathcal{I}$ is a matroid* if and only if *for* any *non-negative modular function f, Algorithm 1 leads to a optimal solution for $\arg\max_{I \in \mathcal{I}} f(I)$.*

Note that Algorithm 1 actually includes many greedy algorithms as special cases (e.g. maximum weighted spanning tree algorithm). The statement of Theorem 2 is so strong that it provides a complete characterization of a large class of problems.

When f is non-decreasing non-negative submodular function, greedy strategy works well for p-system,

Theorem 3 ([10, 40]) *For a non-negative non-decreasing submodular function f, given a p-system $(V, \mathcal{I})$, Algorithm 1 returns a $1/(p+1)$-approximation to the optimal solution.*

This theorem implies immediately that Algorithm 1 gives $1/2$-approximation for matroid constraint.

There are several recent work improving the approximation ratio for matroid constraint (for non-negative monotone submodular functions): based on the idea of continuous greedy process [46] and pipage rounding [1], Calinescu et al. [10] improved the approximation ratio to $(1-1/e)$ (in expectation); Filmus et al. [20] presented a randomized combinatorial $(1-1/e-\epsilon)$-approximation algorithm using only $O(|V|r^3\epsilon^{-3}\log r)$ number of value queries, where r is the size of returned set. Their method is based on local-search and is conceptually much simpler than [10]; Badanidiyuru et al. [3] gave an algorithm that runs in $O(\frac{r|V|}{\epsilon^4}\log^2\frac{r}{\epsilon})$ queries by using a variant of the continuous greedy algorithm.

Non-monotone submodular is normally more difficult to efficiently optimize. We will not discuss in this survey but simply present a summary in Table 1.

3 Applications

In this section, we first show a list of possible applications and discuss several representative applications in detail. We will see from those examples that submodularity is such a natural property that many real-world problems can be cast in to the framework of submodular optimization (maximization).

3.1 List of Possible Applications

- Combinatorial Problems: set cover, max k coverage, vertex cover, edge cover, graph cut problems etc.
- Networks: social networks, viral marketing, diffusion networks etc.
- Graphical Models: image segmentation, tree distributions, factors etc.
- NLP: document summarization, web search, information retrieval
- Machine Learning: active/semi-supervised learning etc.
- Economics: markets, economies of scale

3.2 Classical Problems Revisited

We first show that several well-known problems actually fit into our standard submodular maximization framework.

Exemplar Based Clustering Clustering is one of the most important tasks in the area of data mining. In the k-medoid problem [24] one tries to minimize the sum of pairwise dissimilarities/distances between exemplars and the elements of the dataset. Let $d : V \times V \to \mathbb{R}^+ \cup \{0\}$

be a function that measures the pairwise dissimilarity, we define the k-medoid loss function as following,

$$L(S) = \sum_{e \in V} \min_{v \in S} d(e, v).$$

It is quite straightforward (by Property 1, 5) to show that $-L(S)$ is submodular. By introducing an auxiliary element e_0, we can transform L into a non-negative monotone submodular function,

$$f(S) = L(\{e_0\}) - L(S \cup \{e_0\}).$$

A k-medoid problem can then be formulated as a submodular maximization problem subject to a cardinality constraint,

$$\arg\max_{S \subseteq V : |S| \leq k} f(S).$$

Set Cover Problem The *set cover problem* is an important problem in combinatorial optimization where we are given a collection of subsets of a set E, i.e. $V = \{C_1, C_2, \ldots, C_n\}$ where each $C_i \subseteq E$. We define a function $f : 2^V \to \mathbb{R}$ such that $f(S) = |\cup_{C \in S} C|$. We can interpret f as follow: given S as a subset of V, the value of $f(S)$ is the number of distinct elements covered by the sets in S.

One can easily verify that f satisfies the diminishing return property thus is a submodular function. Furthermore, it is clear that f is non-decreasing. Now given the cardinality constraint, we want to solve the following,

$$\arg\max_{S \subseteq V : |S| \leq k} f(S).$$

We may also assign each $C \in V$ a non-negative cost $w(C)$ (e.g. the size of C), and given a total budget W, our goal is to find a solution of the following,

$$\arg\max_{S \subseteq V : w(S) \leq W} f(S),$$

where $w(S) = \sum_{C \in S} w(C)$. This is a monotone submodular maximization problem under the knapsack constraint.

Maximum Spanning Forest Let us consider a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. In this case we consider E as the ground set and define

$$\mathcal{I} = \{S \subseteq E \mid \text{edge-induced graph } G(V, S) \text{ does not contain a circle}\}.$$

One can verify (via definition) that $(E, \mathcal{I})$ is a matriod. The rank function of $(E, \mathcal{I})$ can be interpreted as the size of the maximum spanning forest of an edge-induced graph, i.e. given $S \subseteq E$, $r(S)$ is the size of maximum spanning forest (in terms of number of edges) of $G(V, S)$.

Now assume that we assign each $e \in E$ a weight $w_e \geq 0$. Let $f : 2^E \to \mathbb{R}$ with $f(S) = \sum_{e \in S} w_e$ be the objective function we want to maximize. Clearly f is monotone and (sub)modular. we consider the following optimization problem,

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

This is exactly the *Maximum Spanning Forest* problem and by Theorem 2, we can solve it efficiently (and exactly) using Algorithm 1.

Maximum Cut in Graphs Given an undirected graph $G = (V, E)$ and a non-negative capacity function $c : E \to \mathbb{R}^+ \cup \{0\}$, the cut capacity function $f : 2^V \to \mathbb{R}$ defined by $f(S) = \sum_{e \in \delta(S)} c(e)$ is submodular, where $\delta(S) = \{e \in E \mid e \text{ has exactly one vertex in } S\}$ i.e. the set of edges crossing S and $E \backslash S$. To shows why f is submodular, we introduce an auxiliary function $f_e : 2^{\{u,v\}} \to \mathbb{R}$ for each $e = \{u, v\} \in E$ which is defined on the graph $G_e = (\{u, v\}, \{e\})$. We define,

- $f_e(\{u, v\}) = f_e(\emptyset) = 0$
- $f_e(\{u\}) = f_e(\{v\}) = w_e$

Then f_e is submodular. We have,

$$f(S) = \sum_{e \in E} f_e(S \cap \{u, v\}).$$

The submodularity of f follows Property 1 and Property 3.

An interesting optimization problem can then be formulated as following,

$$\arg\max_{S \subseteq E : |S| \leq k} f(S).$$

3.3 Applications to NLP

Summarization In the task of summarization, we are given a ground set V and we want to find a subset of V which maximizes some quality measurement under certain constraints. One popular formulation is that, given a function $f : 2^V \to \mathbb{R}$ that measures the quality of a summarization, we try to solve,

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

where $\mathcal{I}$ is a knapsack constraint.

Lin et al. [32] pointed out that a lot of existing work for document summarization task fit the knapsack optimization framework. Furthermore the quality measurement functions being used are usually submodular. Lin et al. also proposed a class of submodular functions that outperform previous work in many aspects. Each of those functions combines two terms, one that encourages the summary to be representative of the corpus, and the other positively rewards diversity. In the task of speech summarization, several submodular functions were discussed by Lin [31]. More discussion on the applications of submodular optimization to summarization can be found in [31].

Word Alignment Word alignment is a key component in most statistical machine translation systems. Unlike classical approaches that utilize graphical models, Lin et al. [33] viewed word alignment problem as submodular maximization problem under matroid constraints.

Suppose that we are given a source language (English) string $e_1, e_2, \ldots, e_n$ and a target language (French) string $f_1, f_2, \ldots, f_m$. Let $V = \{(i, j) \mid i \in [n], j \in [m]\}$ be the ground set. The goal is to find a match $S \subseteq V$ that maximizes a certain quality function under some constraints.

Let $P_1, \ldots, P_m$ be a partition of V where $P_j = [n] \times \{j\}$. The *fertility* restriction of a word f_j then requires that f_j can only match at most k_j words in $e_1, \ldots, e_n$, or equivalently the match $S \subseteq V$ satisfies $|S \cap P_j| \leq k_i$ for all $j \in [m]$. Such a constraint is called *partition constraint*, which is

an important instance of matroid. Lin et al. then proposed a submodular function as the quality function, which is a composition of a concave function and a modular function. We refer readers to [33] for details.

3.4 Applications to Social Networks

Influence Maximization Domingos and Richardson [16, 41] posed a fundamental algorithmic problem: if we can try to convince a subset of individuals (at most k) to adopt a new product or innovation, and the goal is to trigger a large cascade of further adoptions, which set of individuals should we target? Kempe et al. [25] considered such question as a discrete optimization problem. Let $G = (V, E, W)$ a social network (a directed graph with non-negative weights for each edge), two basic models of influence spreading were considered in [25],

- Linear Threshold Mode: each $v \in V$ uniformly at random sample a threshold θ_v from some interval. After initializing $A \subset V$ as the set of active nodes, at each time step, any node v with the total weights of its active neighbors above θ_v will be activated. Repeat this process until no more nodes can be activated.
- Independent Cascade Model: after initializing the active set $A \subset V$, at each time step each active node v will activate its neighbor u (for all inactive neighbors) with probability $w_{v,u}$ (which is the parameter of this network).

Now let $\sigma(A)$ be the expected number of active nodes after long enough time. [25] proved that for both models, $\sigma : 2^V \to \mathbb{Z}$ is a monotone submodular function.

In our application, one is unlikely to able to efficiently compute the exact value of $\sigma(A)$. In fact, one can extend the result of Theorem 1 to show that, by using $(1 + O(\epsilon))$-approximate values for the function to be optimized, we obtain a $(1 - 1/e - \epsilon)$-approximation of the optimum.

Network Structure Inference Gomez Rodriguez et al. [22] first introduced submodular maximization to the context of network structure learning. They considered the problem of learning the network structure in a influence network: given a hidden directed network G^* (vertices are known while edges are not), we observe multiple cascades spreading over it. Each cascade c can be considered as a series of triples. Each triple has the form $(u_i, t_i, \phi_i)_c$ which presents the event that at time t_i, node u_i is reached by c with feature ϕ_i. The goal is to infer the structure of G^* based on a collection of cascades observed (denoted as C).

To model this process, they assumed that in each cascade, each node can only be influenced by at most one other node. So the influence structure of a cascade c is given by a directed tree $T \subseteq G$. Let $P(c|T)$ be the probability of c propagating in a particular tree pattern T, $P(c|G)$ the probability that cascade c occurs in a network G. Let $\mathcal{T}(G)$ be set of all spanning trees of G. [22] then proposed the model

$$P(c|G) = \max_{T \in \mathcal{T}(G)} P(c|T) = \max_{T \in \mathcal{T}(G)} \Pi_{(u,v) \in T} P_c(u, v)$$

where $P_c(u, v) \propto e^{-(t_v - t_u)}$. The structure of the network can be approximated by solving

$$\arg\max_{|G| \leq k} P(C|G) = \arg\max_{|G| \leq k} \Pi_{c \in C} P(c|G).$$

Consider the improvement of log-likelihood for cascade c under graph G over an empty graph K:

$$F_c(G) = \max_{T \in \mathcal{T}(G)} \log P(c|T) - \max_{T \in \mathcal{T}(K)} P(c|T).$$

It turns out that $F_c(G)$ can be expressed as following,

$$F_c(G) = \max_{T \in \mathcal{G}} \sum_{(u,v) \in T} w_c(u, v)$$

where $w_c(u, v)$ is a non-negative weight. Optimizing $P(C|G)$ is equivalently to optimize $F_C(G) = \sum_{c \in C} F_c(G)$ and the latter is shown to be submodular (the ground set is all possible graphs defined over the given vertices set).

3.5 Applications to Machine Learning

Kernel Machines Kernel machines [43] are powerful non-parametric learning techniques. Those approaches use kernel trick to reduce non-linear problems to linear tasks that have been well studied. The data set $V = \{x_1, x_2, \ldots, x_n\}$ is represented in a transformed space via a kernel matrix

$$K_V = \begin{pmatrix} \mathcal{K}(x_1, x_2) & \ldots & \mathcal{K}(x_1, x_n) \\ \vdots & \ddots & \vdots \\ \mathcal{K}(x_n, x_1) & \ldots & \mathcal{K}(x_n, x_n) \end{pmatrix}$$

here $\mathcal{K} : V \times V \to \mathbb{R}$ is the kernel function that is symmetric and positive definite.

For large-scale problem, even representing the matrix K_V (which requires $O(n^2)$ space) is prohibited. The common solution to solve this issue is to select a small representative subset $S \subseteq V$ and only work with K_S. An additional benefit by using such a subset is that it may help to avoid overfitting. The question now becomes how to select the subset S. One popular way to measure the quality of selected set S is to use *Informative Vector Machine*(IVM) introduced by Laurence et al. [29]. Formally, we define $f : 2^V \to \mathbb{R}$ with

$$f(S) = \frac{1}{2} \log \det \left(\mathbf{I} + \sigma^{-2} K_S \right)$$

where $\mathbf{I}$ is the identity matrix and $\sigma > 0$ is a parameter. IVM has close connection with the entropy of muti-variable Gaussian distribution [5] and it is shown in [5, 44] that f is a submodular function. We can then select the set $S \subset V$ by solving

$$\arg\max_{S : |S| \leq k} f(S).$$

3.6 More Applications

- [35] distributed k-centers using submodular optimization.

constraint	monotone	non-negative
cardinality	$\frac{1-\epsilon}{2}$ [2]	$\frac{1-\epsilon}{2+e}$ [12], R
matroid	$1/4$ [11]	$\frac{1-\epsilon}{4+e}$ [12], R
matching	$4/31$ [11]	$\frac{1-\epsilon}{12+\epsilon}$ [12], R
intersection of p matroids	$\frac{1}{4p}$ [11]	$\frac{(1-\epsilon)(p-1)}{5p^2-4p+\epsilon}$ [12], R
p-matchoid	$\frac{1}{4p}$ [12]	$\frac{(1-\epsilon)(2-o(1))}{(8+e)p}$ [12], R

Table 2: Best known approximation bounds for submodular maximization in streaming model. Bounds for randomized algorithms that hold in expectation are marked (R).

4 Streaming Submodular Maximization

In this section, we cover recent results for submodular maximization over data streaming. In the streaming model, we consider the ground set V as an ordered sequence of items $e_1, e_2, \ldots, e_n$. We process the items one by one and the maximum space being used should be sublinear (i.e. $o(n)$).

Unlike in centralized setting where one can always calculate the function value, in streaming model we need to assume value oracle and membership oracle explicitly. Most algorithms developed for centralized submodular maximization require access to all items in the ground set and hence can not be directly applied in streaming setting. In general streaming submodular maximization is harder and was considered only very recently. We provide a table for best known approximation bounds in Table 2 and review some technical details of those related papers.

4.1 Results for Cardinality Constraint

Monotone submodular maximization with cardinality constraint is the simplest setting and it is most likely to have a good solution. Krause et al. [27] gave the first discussion on maximizing a monotone submodular function over data streams. Their approach makes strong assumption on the input data and the update time per item is as high as $O(k)$ value queries (k is the size of the solution). Kumar et al. [28] proposed a single-pass which assumes an upper bound on $\max_{e \in V} \Delta(e|\emptyset)$. [28] also includes a multi-pass algorithm with space usage depending on the ground set. Badanidiyuru et al. [2] provided a nice solution to monotone submodular maximization under cardinality constraint. They also discussed the situation when value oracle is not realistic in streaming model.

As for non-monotone submodular maximization, Buchbinder et al. [9] gaves a .0893-approximation, and the very general framework of Chekuri et al. [12] yields a $\frac{1-\epsilon}{2+e}$-approximation (in expectation).

CHEN : *add a table for streaming results*

Monotone Case: We only describe [2], which is the best known result in many aspects. In Algorithm 1, it is known that the marginal gain $\Delta(e_{i+1}|S_i)$ of the next element e_{i+1} added is at least $(\text{OPT} - f(S_i))/k$ where $\text{OPT} = \max_{S:|S|\leq k} f(S)$, and S_i is the set of the first i elements picked. This intuition also helps in streaming setting: if we know OPT in advance, we can use $\frac{\text{OPT}/2 - f(S)}{k-|S|}$ as a threshold of marginal gain to decide if a new element e should be added to the solution S or not (denote such an algorithm as KNOWOPT). Of course we do not know OPT, but for any $e \in V$, we must have $m \leq \text{OPT} \leq k \cdot m$ where $m = \max_{e \in V} f(\{e\})$. So we can guess the value of OPT

using values in $\{(1+\epsilon)^i \mid i \in \mathbb{Z}, m \leq (1+\epsilon)^i \leq k \cdot m\}$ and for each guess we run an instance of KNOWOPT. Such an algorithm, however, requires two passes with m being decided in the first pass. With the trick of lazy-evaluation, we can actually estimate m on the fly. The final algorithm can is depicted in Algorithm 2.

Algorithm 2: SIEVE-STREAMING for submodular maximization

Input: V as data stream, f a monotone submodular function, k the size constraint, ϵ a parameter
Output: a set $S \subseteq V$

$O = \{(1+\epsilon)^i \mid i \in \mathbb{Z}\}$
`/* maintain the sets only for the necessary v's lazily */`
For each $v \in O,\ S_v \leftarrow \emptyset$
$m \leftarrow 0$
for *each* e *in the data stream* **do**
    m $\leftarrow \max\{m, f(\{e\})\}$
    $O \leftarrow \{(1+\epsilon)^i \mid m \leq (1+\epsilon)^i \leq 2 \cdot k \cdot m\}$
    Delete all S_v such that $v \in O$
    for $v \in O$ **do**
        if $\Delta(e|S_v) \geq \frac{v/2 - f(S_v)}{k - |S_v|}$ *and* $|S_v| < k$ **then**
            $S_v \leftarrow S_v \cup \{e\}$
return $\arg\max_{S_v : v \in O} f(S_v)$

It can be shown that Algorithm 2 achieves $(1/2 - \epsilon)$-approximation by using $O(\frac{k \log k}{\epsilon})$ space and $O(\frac{\log k}{\epsilon})$ update time per item.

Non-negative Non-Monotone Case: Non-negative non-monotone case can be much harder. The state of the art comes from [12]. The algorithm we discuss here is an instance of their general framework and it goes as follows: the algorithm is parameterized with a threshold $\alpha > 0$ and maintains a buffer B of size $\frac{k}{\epsilon}$; the algorithm then keeps every new coming item with marginal gain above the threshold into the buffer, until the buffer is overflowed; an item e is then randomly sampled from B and added to the solution S; all other items with marginal gain below the threshold will be removed from B. We describe the detail in Algorithm 3.

Algorithm 3: RANDOM-STREAMING for non-monotone submodular maximization

Input: V as data stream, f a non-negative submodular function, k the size constraint, ϵ a parameter

Output: a set $S \subseteq V$

```
B ← ∅, S ← ∅
for each e in the data stream do
    if |S| < k and Δ(e|S) > α then
        B ← B + e
    if |B| > k/ε then
        e ← uniformly random from B
        B ← B − e, S ← S + e
        for all e' ∈ B s.t. Δ(e'|S) ≤ α do
            B ← B − e'
S' ← offline algorithm on B
return arg max_{A∈{S,S'}} f(A)
```

When taking $\frac{1-\epsilon}{(2+\epsilon)k}\text{OPT} \leq \alpha \leq \frac{1+\epsilon}{(2+e)k}\text{OPT}$, Algorithm 3 yields a $\frac{1-\epsilon}{2+\epsilon}$-approximation. Of course we do not know OPT, but the same trick in SIEVE-STREAMING also applies here. The space usage is bounded by $O(\frac{k \log k}{\epsilon})$.

4.2 Results for Matroids and Its Intersection

For general non-negative submodular maximization subject to matroid constraint, the general framework in [12] yields a $\frac{1-\epsilon}{4+\epsilon}$-approximation (which we will not discuss in detail). For monotone case, a better result was given by Chakrabarti and Kale [11], which follows the algorithm of Varadaraja [45] on modular maximization. In this section, we mainly discuss the results from [11, 45].

Varadaraja [45] proposed an algorithm for normalized modular maximization subject to the intersection of p matroids (it actually works for more general problems, be we only need its result for modular function). The algorithm is very simple and we would like to describe here as pseudocode. Also recall that, for a normalized modular function f, we have $f(S) = \sum_{e \in S} f(\{e\})$.

Algorithm 4: Streaming algorithm for non-decreasing modular function

Input: V as data stream, f a non-decreasing modular function, p matroid constraints $\mathcal{I}_1, \ldots, \mathcal{I}_p$, $\gamma > 0$ is a parameter
Output: a set $S \subseteq V$

```
for each e in the data stream do
    if S + e ∈ ∩_i I_i then
        S ← S + e
    else
        C ← ∅
        for each I_i do
            if S + e ∉ I_i then
                e_i ← arg min_{e' : S+e−e' ∈ I_i} f({e'})
                C ← C ∪ e_i
            if f({e}) ≥ (1 + γ) · (Σ_{e'∈C} f({e'})) then
                S ← S ∪ {e}\C
return S
```

Varadaraja proved that Algorithm 4 returns a $\frac{\gamma}{(p\cdot(1+\gamma)-1)(1+\gamma)}$-approximation. In particular, this algorithm returns a $(1+\gamma)$-approximation for matroid constraint. For convenience, we also call the non-decreasing normalized modular maximization problem as *maximum weighted* (MW) problem.

Chakrabarti et al. [11] proposed a general framework called *compliant* algorithm. A key theorem in [11] then shows that if a compliant algorithm for MW has some approximation guarantee, running the same algorithm for normalized monotone submodular also gives certain guarantee, informally,

Theorem 4 *If a compliant algorithm for MW runs on parameter γ and returns C_γ-approximation, running the same algorithm for normalized monotone submodular function gives $(C_\gamma+1+1/\gamma)$-approximation.*

As expected, Algorithm 4 is compliant with parameter γ. By taking $\gamma = 1$, we obtain $\frac{1}{4p}$-approximation for p matroids intersection and $1/4$-approximation for matroid constraint.

4.3 Results for Matching and p-matchoid

For matching constraint, Chakrabarti et al. [11] noted that the algorithm in [49] for streaming graph matching is a compliant algorithm. By applying Theorem 4, they obtained an algorithm for normalized monotone submodular with $\frac{4}{31}$-approximation guarantee.

[12] is the only paper that gives some results on p-matchoid. We will not discuss its technical details but summarize its results in Table 2.

5 Distributed Submodular Maximization

As a special case of the more general distributed submodular maximization, the distributed set cover problem (see Sec. 3.2) was discussed in [13]. A polylog-round MapReduce algorithm with

$(1 - e^{-1})$-approximation guarantee was proposed in that paper. The number of rounds was later reduced to $O(\log^2 |V|)$ by Blelloch et al. [6].

The general distributed submodular maximization problem was systematically studied by Kumar et al. [28] in MapReduce model. For the cardinality constraint, Mirzasoleiman et al. [39] proposed a simple two-round MapReduce algorithm that performs well in practice. Barbosa et al. [14] provided a more sophisticated analysis for a randomized version of the algorithm in [39], and provided much stronger theoretical guarantee. Those results were further extended in [4]. Inspired by [39], Mirrokni and Zadimoghaddam [37] provided a general framework for distributed submodular maximization using the concept of randomized composable core-sets. In this section, we mainly discuss the results from [4, 14, 28, 37, 39].

Unlike in the streaming where an algorithm is normally designed from scratch, in the distributed setting, centralized algorithms are usually used as black boxes. For this reason, our discussion focus more on the theory of the framework, i.e. the relation between the guarantee in the distributed setting with the guarantee of the centralized algorithm(s) being used.

5.1 MapReduce Model

Most distributed submodular maximization algorithms are discussed under the *MapReduce* model [15] which we briefly describe here.

In the MapReduce model, the data is represented as ⟨key, value⟩ pairs that are distributed across m machines. A computation in this model proceeds in rounds. In each round, there will be two phase.

- Map phase: each pair ⟨key, value⟩ is mapped by a user-defined hash function to ⟨hash(key), value⟩; all pairs are then shuffled and sent to different machines, and it is guaranteed that all pairs with the same hash value will be distributed to the same machine.
- Reduce phase: each machine performs computation on the pairs it received as the output or the input of the next round.

All distributed algorithms we cover are discussed in the MapReduce model.

5.2 Algorithms via Sample & Prune

The techniques being used by Kumar et al. [28] are very different from other ideas we will cover. Only monotone submodular functions were considered in their paper. There are two main ideas in their framework,

- the approximated version of Algorithm 1 (called ϵ-GREEDY) where in each step, the algorithm only tries to find $e \in V \backslash S$ with marginal gain $(1 - \epsilon)$ to the optimum; it can be shown that this algorithm performs almost as well as Algorithm 1
- a Sample&Prune step: in each round the algorithm sample a subset from the remaining elements and run greedy algorithm on that set to obtain a immediate solution; the immediate solution is then used to prune those impossible to be part of the final solution

The framework then uses the Sample&Prune technique to simulate the ϵ-GREEDY algorithm. Let $\Delta = \max_{e \in V} \Delta(e|\emptyset)$. They shown that for hereditary constraint, their framework simulates

the ϵ-GREEDY using $O(\frac{1}{\epsilon\delta}\log\Delta)$ rounds of MapReduce. $O(n\log n/\mu)$ machines are used and each machine use $\mu = O(kn^{\delta}\log n)$ space. Here k is the size of returned solution.

It is shown in [10] that the ϵ-GREEDY algorithm achieves: 1) $\frac{1}{2+\epsilon}$-approximation for matroid constraint; 2) $\frac{1-e^{-1}}{1+\epsilon}$-approximation for cardinality constraint; 3) $\frac{1}{p+1+\epsilon}$-approximation for p-system.

Those are guarantees can be achieved by the framework in Kumar et al. [28] as well.

5.3 Theory for GREEDI-Based Algorithms

GREEDI is a simple MapReduce algorithm proposed by Mirzasoleiman et al. [39] for monotone submodular maximization under cardinality constraint. Based on this algorithm, [14, 37] gave deep theoretical results. The GREEDI algorithm proceeds in two rounds:

1. partition the ground set into m parts and for each part V_i run the standard greedy algorithm (Algorithm 1) with constraint k to obtain a corresponding solution S_i
2. union all solutions obtained in the first round and run Algorithm 1 with constraint k to obtain a global solution S, return the best among S_i's and S

Unfortunately, the approximation guarantee given by this algorithm is only bounded by $(1-e^{-1})^2/\min(m,k)$. Mirzasoleiman et al. [39] also gave better bound for datasets with certain geometric structure.

Noting that the method in [39] gives excellent performance in practice, Barbosa et al [14] conducted a more sophisticate analysis. Their results show that if the ground set V is randomly partitioned to m machines, one can achieve significantly better approximation guarantee, namely, $\frac{1-e^{-1}}{2}$ (in expectation) for monotone submodular maximization subject to cardinality constraint . In fact, Barbosa et al. [14] proved much stronger results which we would like to summarize here as an informal theorem,

Theorem 5 ([14], informal) *Let f be a sumbodular function and let $\mathcal{I}$ be a hereditary constraint. Assume* ALG *is an algorithm that gives α-approximation for maximizing f subject to $\mathcal{I}$. We do random partition in* GREEDI *and replace standard greedy algorithm with* ALG*, the constraint we consider here is $\mathcal{I}$ instead of the cardinality constraint. Then*

- *if f is monotone, above algorithm gives $\frac{\alpha}{2}$-approximation*
- *if f is non-monotone and* ALG *satisfies a certain technical property (see [14] for detail), above algorithm gives $\frac{\alpha}{4+2\alpha}$-approximation.*

Note that the only requirement for the constraint is *hereditary*, so the theorem holds for matroid, knapsack, and p-system.

Mirrokni et al. [37] analyzed a class of GREEDI-like algorithms using the concept of randomized composable core-sets. In their framework each item in the ground set V can be randomly assigned to C machines. Machine i was assigned $T_i \subseteq V$. Also we define f_k be the function that takes V returns an solution to $\arg\max_{S:|S|\leq k} f(S)$. The randomized composable core-sets is defined as follows,

Definition 9 (Informal) Consider an algorithm ALG that given any subset $T \subseteq V$ returns a subset $\text{ALG}(T) \subseteq T$ with size at most k'. Let $\{T_1, \ldots, T_m\}$ generated as desired above, if

$$\mathbf{E}\left[f_k(\text{ALG}(T_1) \cup \ldots \cup \text{ALG}(T_m))\right] \geq \alpha \cdot \mathbf{E}\left[f_k(T_1 \cup \ldots \cup T_m)\right],$$

we call ALG is an α-approximate randomized composable core-sets with parameter C, k', k, or simply α-core-sets.

If an algorithm is α-core-sets and the corresponding f_k is always gives β-approximation for a submodular maximization problem, we directly obtain a distributed algorithm that gives $\alpha \cdot \beta$-approximation.

Mirrokni et al. [37] then proved that a class of classical algorithm are actually α-core-sets (for some α). Moreover, if one is allowed to increase k' (recall that $\text{ALG}(T)$ return a set with size k'), one may obtain better approximation guarantee. Increasing C (recall that each $e \in V$ is randomly assigned to C machines) also helps to increase the approximation ratio. We would not cover the detailed theorem, but summarize some of them in Table 3.

5.3.1 Other Results

More recent work of Barbosa et al. [4] proposed a different framework that runs in $O(1/\epsilon)$ rounds. The algorithm is somewhat involved so we ignore its details here. For an algorithm ALG designed for submodular maximization subject to hereditary constraint, Barbosa et al introduced two properties,

- α-approximation: let $T = \arg\max_{S \in \mathcal{I}} f(S)$, for any $S \subseteq V$

$$f(\text{ALG}(S)) \geq \alpha \cdot f(T \cap S)$$

- consistency: for two disjoint sets $A, B \subseteq V$, if for every $e \in B$ we have $\text{ALG}(A \cup \{e\}) = \text{ALG}(A)$, then $\text{ALG}(A \cup B) = \text{ALG}(A)$

It turns out any deterministic algorithm with those two properties, can be adapted to a distributed algorithm that runs in $O(1/\epsilon)$ rounds and gives $(\alpha - \epsilon)$-approximation.

Finally, we summarize some of the best results known in Table 3.

6 Conclusion and Future Research

References

[1] A. A. Ageev and M. I. Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8(3):307–328, 2004.

[2] 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*, pages 671–680. ACM, 2014.

constraint	rounds	approx.	reference
cardinality	$O(\frac{\log n}{\epsilon})$	$1 - e^{-1} - \epsilon$	[28]
	2	0.545	[37]
	$O(1/\epsilon)$	$1 - e^{-1} - \epsilon$	[4]
matroid	$O(\frac{\log n}{\epsilon})$	$1/2 - \epsilon$	[28]
	2	$1/4$	[14]
	$O(1/\epsilon)$	$1 - e^{-1} - \epsilon$	[4]
p-system	$O(\frac{\log n}{\epsilon})$	$\frac{1}{p+1} - \epsilon$	[28]
	2	$\frac{1}{2(p+1)}$	[14]
	$O(1/\epsilon)$	$\frac{1}{p+1} - \epsilon$	[4]

Table 3: Best known algorithms for monotone submodular maximization in the MapReduce model. All algorithms are randomized.

[3] A. Badanidiyuru and J. Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1497–1514. SIAM, 2014.

[4] R. d. P. Barbosa, A. Ene, H. L. Nguyen, and J. Ward. A new framework for distributed submodular maximization. *arXiv preprint arXiv:1507.03719*, 2015.

[5] J. Bilmes. Lecture notes on submodular optimization. `http://j.ee.washington.edu/~bilmes/classes/ee596b_spring_2014/`, 2014.

[6] G. E. Blelloch, H. V. Simhadri, and K. Tangwongsan. Parallel and i/o efficient set covering algorithms. In *Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures*, pages 82–90. ACM, 2012.

[7] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE, 2001.

[8] 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*, pages 1433–1452. SIAM, 2014.

[9] N. Buchbinder, M. Feldman, and R. Schwartz. Online submodular maximization with preemption. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1202–1216. SIAM, 2015.

[10] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

[11] A. Chakrabarti and S. Kale. Submodular maximization meets streaming: Matchings, matroids, and more. *Mathematical Programming*, 154(1-2):225–247, 2015.

[12] C. Chekuri, S. Gupta, and K. Quanrud. Streaming algorithms for submodular function maximization. *arXiv preprint arXiv:1504.08024*, 2015.

[13] F. Chierichetti, R. Kumar, and A. Tomkins. Max-cover in map-reduce. In *Proceedings of the 19th international conference on World wide web*, pages 231–240. ACM, 2010.

[14] R. da Ponte Barbosa, A. Ene, H. L. Nguyen, and J. Ward. The power of randomization: Distributed submodular maximization on massive datasets. *arXiv preprint arXiv:1502.02606*, 2015.

[15] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[16] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–66. ACM, 2001.

[17] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[18] M. Feldman, J. Naor, and R. Schwartz. A unified continuous greedy algorithm for submodular maximization. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 570–579. IEEE, 2011.

[19] M. Feldman, J. S. Naor, R. Schwartz, and J. Ward. Improved approximations for k-exchange systems. In *Algorithms–ESA 2011*, pages 784–798. Springer, 2011.

[20] Y. Filmus and J. Ward. Monotone submodular maximization over a matroid via non-oblivious local search. *SIAM Journal on Computing*, 43(2):514–542, 2014.

[21] S. Fujishige. *Submodular functions and optimization*, volume 58. Elsevier, 2005.

[22] 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*, pages 1019–1028. ACM, 2010.

[23] T. A. Jenkyns. The efficacy of the greedy algorithm. In *Proceedings of the 7th Southeastern Conference on Combinatorics, Graph Theory, and Computing, Utilitas Mathematica, Winnipeg*, pages 341–350, 1976.

[24] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[25] D. Kempe, J. Kleinberg, and É. 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*, pages 137–146. ACM, 2003.

[26] P. Kohli, M. P. Kumar, and P. H. Torr. P^3 & beyond: Move making algorithms for solving higher order functions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(9):1645–1656, 2009.

[27] A. Krause and R. G. Gomes. Budgeted nonparametric learning from data streams. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 391–398, 2010.

[28] R. Kumar, B. Moseley, S. Vassilvitskii, and A. Vattani. Fast greedy algorithms in mapreduce and streaming. *ACM Transactions on Parallel Computing*, 2(3):14, 2015.

[29] N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse gaussian process methods: The informative vector machine. In *Proceedings of the 16th Annual Conference on Neural Information Processing Systems*, number EPFL-CONF-161319, pages 609–616, 2003.

[30] J. Lee, M. Sviridenko, and J. Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Mathematics of Operations Research*, 35(4):795–806, 2010.

[31] H. Lin. *Submodularity in Natural Language Processing: Algorithms and Applications*. PhD thesis, 2012.

[32] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics, 2011.

[33] H. Lin and J. Bilmes. Word alignment via submodular maximization over matroids. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 170–175. Association for Computational Linguistics, 2011.

[34] L. Lovász. Submodular functions and convexity. In *Mathematical Programming The State of the Art*, pages 235–257. Springer, 1983.

[35] G. Malkomes, M. J. Kusner, W. Chen, K. Q. Weinberger, and B. Moseley. Fast distributed k-center clustering with outliers on massive data. In *Advances in Neural Information Processing Systems*, pages 1063–1071, 2015.

[36] M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, pages 234–243. Springer, 1978.

[37] V. Mirrokni and M. Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. *arXiv preprint arXiv:1506.06715*, 2015.

[38] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrak, and A. Krause. Lazier than lazy greedy. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[39] 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*, pages 2049–2057, 2013.

[40] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

[41] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70. ACM, 2002.

[42] A. D. Sarma, A. Lall, D. Nanongkai, R. J. Lipton, and J. Xu. Representative skylines using threshold-based preference distributions. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 387–398. IEEE, 2011.

[43] B. Schölkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[44] M. Seeger. Greedy forward selection in the informative vector machine. Technical report, Technical report, University of California at Berkeley, 2004.

[45] A. B. Varadaraja. Buyback problem-approximate matroid intersection with cancellation costs. In *Automata, Languages and Programming*, pages 379–390. Springer, 2011.

[46] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 67–74. ACM, 2008.

[47] J. Vondrák, C. Chekuri, and R. Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 783–792. ACM, 2011.

[48] K. Wei, R. Iyer, and J. Bilmes. Fast multi-stage submodular maximization. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1494–1502, 2014.

[49] M. Zelke. Weighted matching in the semi-streaming model. *Algorithmica*, 62(1-2):1–20, 2012.