Submodular Function Maximization

Kaixuan Hu

April 2019

1 Introduction

Optimization problems involving submodular function subject to different kinds of constraints are very common in both theory and practice and it has rich and varied applications in different area. It can be thought as the discrete analogues of concave or convex functions. The problem of maximizing a submodular function is NP-hard, and there has been extensive research into finding a good approximation to this problem.

There are many different settings of the problem, for instance, we are interested in studying the problem under some constraints such as cardinality constraints, knapsack constraints, matroid constraints, and for the cases where the objective function f is monotone or non-monotone. The main part of the paper will be devoted to introduce a series of works on improving the approximation ratio for maximizing a non-monotone submodular function subject to a matroid constraint. And it shall serve as a starting point for further exploring the problem of submodular function maximization.

2 Problem Description

Definition 1. Let U be a universe of elements (i.e. the *ground set*), and 2^U denotes the collection of all subsets of U.

A set function $f : 2^U \to \mathbb{R}$ is *submodular* if for every $A, B \subseteq U$, we have:

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

f is *monotone* if for every $A \subseteq B$, we have $f(A) \leq f(B)$.

We further assume $f(\varnothing) = 0$ for the purpose of simplifying the analysis and proofs. We use the *value oracle* model, for which we can evaluate the value of $f(S)$ in constant time for any $S \subseteq U$.

The problem we are interested in is to find a subset S of U that $f(S)$ is maximized with no constraint or subjecting to some constraint. In the unconstrained setting (USM), we learned in this class that there is a well-known deterministic and a randomized "double-sided" online algorithm of Buchbinder *et al.* [3] that provide a deterministic $1/3$ approximation and randomized $1/2$ approximation to the problem of USM.

One of the most commonly used and well studied constraint is called the *matroid constraint*, which is a generalization of the *cardinality constraint* (i.e. maximize $f(S)$ subject to $|S| \leq k$, for some integer $k \leq |U|$), we first give the definition of matroid:

Definition 2. Let U be the universe and $\mathcal{F}$ be a family or collection of the subsets of U, a pair $\mathcal{M} = (U, \mathcal{F})$ is a *matroid* if the following is satisfied:

1. $\varnothing \in \mathcal{F}$
2. $\forall A \subseteq B; B \in \mathcal{F}$ implies $A \in \mathcal{F}$ (*Down-closed* property).
3. If $A, B \in \mathcal{F}$, and $|A| > |B|$, then there is $x \in A \setminus B$ such that $B \cup \{x\} \in \mathcal{F}$.

And we call the element $I \in \mathcal{F}$ an independent set.

Under the matroid constraint, let $\mathcal{M} = (U, \mathcal{F})$ be a matroid, we want to maximize $f(S)$ subject to $S \in \mathcal{F}$.

When subjecting to a constant number of knapsack constraints (i.e. $O(1)$-knapsack constraints), we are given a ground set U and a set of d knapsack constraints over U, we want to find a set $S \subseteq U$ satisfying all d knapsack constraints and maximizing $f(S)$. For instance, if we have only one knapsack with capacity C and each element $e \in U$ has some cost $c(e) \geq 0$, then the objective is: $\max_S f(S)$ subject to $\sum_{e \in S} c(e) \leq C$.

3 Measured Continuous Greedy

A common approach to the problem of constrained submodular maximization is based on two main steps. The first step is, instead of maximizing $f(S), S \subseteq U$ directly, we are trying to find a fractional solution that maximizes the *multilinear extension* to f. Then the second step is to apply some rounding technique to the fractional solution and yields some integral solution.

The work of Feldman *et al.* [1] mainly focus on improving the first step, which is finding a fractional solution for the relaxation of the problem, and gave a *Measured Continuous Greedy algorithm* that achieves an approximation ratio of $(1 - 1/e)$ for the monotone case, which matches the previously known result. And more importantly it achieves an improved approximation ratio of about $1/e$ for the general non-monotone case.

A very important concept involved in this framework as well as the two steps approach to the submodular maximization problem is the *multilinear extension* of the submodular function, introduced by the work of Calinescu *et al.* [4] and Vondrák [2].

3.1 Multilinear Extension

Let f be a set function, to talk about the multilinear extension of a submodular function, it is helpful to define f in an equivalent way:

$$f : \{0,1\}^n \to \mathbb{R}_+$$

where $n = |U|$. Essentially, f takes in an indicator vector representing a subset of the universe U, if the subset contains the i^{th} element, then the i^{th} entry of the vector is 1, else, the i^{th} entry of 0.

The *Multilinear Extension* F of the function f is a continuous function that extends f from over the corners of the unit hyper-cube $\{0,1\}^n$ to the entire unit hyper-cube $[0,1]^U$, i.e. $F : [0,1]^U \to \mathbb{R}_+$, and F represents the expected value of f over a random subset $R(x) \subseteq U$. Essentially, F maps a vector of probabilities, where each entry x_i represents the probability of the element $i \in U$ being independently included in $R(x)$, to the expected value defined as:

$$\begin{aligned} F(\mathbf{x}) &= \mathbb{E}[f(R(x))] \\ &= \sum_{S' \subseteq U} f(S') \prod_{i \in S'} x_i \prod_{j \notin S'} (1 - x_j) \end{aligned}$$

As mentioned above, a recent widely used approach to the problem of maximizing $f(S)$ is to first maximize its multilinear extension $F(\mathbf{x})$ where $\mathbf{x}$ is in some $P \subseteq [0,1]^n$, P is determined by the constraints of the problem. In the case of matroid constraints, given a matroid $\mathcal{M} = (U, \mathcal{F})$, P specifies a set of linear constraints which define a down-closed or down-monotone *matroid polytope* that is the convex hull of the independent sets $I \in \mathcal{F}$.

We now have some fractional solution from the first step, then the second step is to round the fractional solution $\mathbf{x}$ to some integer solution in $\mathcal{F} \subseteq 2^U$. Chekuri *et al* [5]. gave the *contention resolution framework* that provides a general method for this step.

This two-step approach is introduced by [2, 4] along with a *continuous greedy* algorithm for computing a fractional solution to the problem of submodular function maximization subject to matroid constraints.

3.2 Algorithm Description

The work of Feldman *et al.* [1] focus on improving the first step of the continuous greedy approach by trying to give a generalization to the continuous greedy algorithm, and presented a *Measured Continuous Greedy algorithm* that improves the approximation ratio for the general non-monotone case to $1/e$ subject to the matroid constraint, compare to the previously result of approximately ≈ 0.325 [5]. Their contribution consists of two main parts.

3.2.1 Distorting the Algorithm's Direction

Firstly, we briefly describe the continuous greedy algorithm [2, 4]. The algorithm is essentially following a continuous greedy process. It keeps track of a particle parameterized by a function $\mathbf{y} : [0, 1] \to P$, mapping from a unit time interval to P (which is the *matroid polytope* introduced above). The particle starts at $\mathbf{y}(0) = \mathbf{0}$ and at each step, the algorithm moves the particle by a small amount ϵ in some direction z chosen from $z \in P$.

And z is chosen by solving:

$$z^* = \operatorname*{argmax}_{z \in P} \big(\langle z, \nabla F(\mathbf{y}) \rangle\big)$$

at each iteration. (We can view z as $z = \frac{d\mathbf{y}}{dt}$, note that $\langle\ ,\ \rangle$ denotes the vector dot product and also note that we use z instead of $\mathbf{z}$ to denote a direction vector throughout the text).

At the end of the process, i.e. at time $t = 1$, by the above method of choosing the moving direction of the particle, we have:

$$F(\mathbf{y}(1)) \geq (1 - 1/e) \cdot F(OPT)$$

where OPT is the optimal integral solution. i.e. $\mathbf{y}(1)$ is the position that yields a $(1 - 1/e)$ approximation of maximizing the multilinear extension F over the matroid polytope P.

Notice that for the gradient of the multilinear extension F at $\mathbf{y}$, i.e. $\nabla F(\mathbf{y})$, we don't use it in the implementation of the algorithm, instead the continuous greedy algorithm approximates or mimics it by the *residual increase* of each element $i \in U$, i.e. $F(\mathbf{y} \vee \mathbf{1}_i) - F(\mathbf{y})$, where $\mathbf{1}_i$ is the indicator vector of the set $\{i\}$, and $(\mathbf{y} \vee \mathbf{1}_i)$ is the element-wise maximum vector, i.e. $(\mathbf{y} \vee \mathbf{1}_i)_e = \max\{\mathbf{y}_e, (\mathbf{1}_i)_e\}$.

In order to get a better approximation of the $\nabla F(\mathbf{y})$ than the residual increase of elements at position $\mathbf{y}$, the measured continuous greedy algorithm *distorting* the moving direction z by decreasing z_i for each $i \in U$ by a multiplicative factor of $(1 - \mathbf{y}_i)$, where $\mathbf{y}$ is the current location of the particle. i.e. the main idea of the algorithm is to first solve:

$$z^* = \operatorname*{argmax}_{z \in P} \big(\langle z, w(\mathbf{y}) \rangle\big)$$

where $w(\mathbf{y}) \in \mathbb{R}^U$, and $(w(\mathbf{y}))_i = F(\mathbf{y} \vee \mathbf{1}_i) - F(\mathbf{y})$ is the residual increase for each $i \in U$. The $\langle z, w(\mathbf{y}) \rangle$ can be viewed as the *rate of change* of $F(\mathbf{y})$.

After that, as mentioned above, we multiply each entry of the direction z_i^* by $(1 - \mathbf{y}_i)$ for each $i \in U$ as the new moving direction of the particle.

We rewrite the procedure formally, given time step t, suppose the current position of the algorithm is $\mathbf{y}$,

1. For each $i \in U$: compute $(w(\mathbf{y}))_i = F(\mathbf{y} \vee \mathbf{1}_i) - F(\mathbf{y})$
2. Compute $z^* = \operatorname{argmax}_{z \in P} \big(\langle z, w(\mathbf{y}) \rangle\big)$
3. For each $i \in U$: increase each entry i of the position $\mathbf{y}$ by $\delta \cdot z_i^* \cdot (1 - \mathbf{y}_i)$

This is a simplified version of the Algorithm 1 from [1], and δ is some constant.

3.2.2 The Stopping Time Parameter

Secondly, they introduced a new parameter called *stopping time* in the algorithm. The improvement that the stopping time parameter brings in is an extension to the *contention resolution framework* [5] mentioned above. We won't go into the detail of the framework, but in summary, the introduction of the stopping time parameter simplifies the framework and combines the step of finding the fractional solution and the step of re-normalization (this is involved in the process of rounding the fractional solution). And it also allows some controls over the quality of the fractional solution that can be described by the following results from [1]:

For the case where we have a general submodular function f under the matroid constraint:

$$F(\mathbf{y}) \geq [Te^{-T} - o(1)] \cdot f(OPT) \text{ where } \mathbf{y}/T \in P$$

And for the case where we have a monotone submodular function f under the matroid constraint:

$$F(\mathbf{y}) \geq [1 - e^{-T} - o(1)] \cdot f(OPT) \text{ where } \mathbf{y}/T \in P$$

3.3 Pipage Rounding

After we obtain a fractional solution $\mathbf{y}(1) \in P$ to the problem of maximization of the multilinear extension F over the polytope P, we can efficiently round $\mathbf{y}(1)$ to some discrete solution $S \in \mathcal{F}$ without too much loss in the objective function value. And the technique is called *pipage rounding* [7].

Theorem 1. (Theorem 2.3. in [1]) Given a matroid $\mathcal{M} = (U, \mathcal{F})$, a submodular function $f : 2^U \to \mathbb{R}_+$ with its multilinear extension $F : [0,1]^U \to \mathbb{R}_+$ and a point $\mathbf{x}$ in the matroid polytope $P(\mathcal{M})$, there is a polynomial time algorithm, called pipage rounding, returning a random independent set $S \in \mathcal{F}$ such that $\mathbb{E}[f(S)] \geq F(\mathbf{x})$.

3.4 Main Result of the Algorithm

Combining the work in section 3.2 and 3.3, we have one of the most important results in [1].

Theorem 2. (Theorem 4.1. in [1]) Given a matroid $\mathcal{M} = (U, \mathcal{F})$ and a submodular function $f : 2^U \to \mathbb{R}_+$, there is a polynomial time $(1/e - o(1))$-approximation algorithm for finding an independent set $S \in \mathcal{F}$ to maximize $f(S)$ (i.e. subject to a matroid constraint).

3.5 Further Improvement on the Algorithm

3.5.1 The First Improvement

For the problem of maximizing a non-monotone submodular function subject to matroid constraints. The measured continuous greedy algorithm [1] brings an improvement to the approximation ratio from 0.325 [5] to $1/e$. However, the $1/e$ approximation ratio is still not optimal. The work of Ene and Nguyen [6] further improved this ratio to 0.372 and it is the first improvement since [1].

It is worth noting that the notation for the *rate of change* of $F(\mathbf{y})$ is different in the two paper [1] and [6]. In [1], the *residual increase* of each element e is used, i.e. $F(\mathbf{y} \vee \mathbf{1}_e) - F(\mathbf{y}) = w_e(\mathbf{y})$, and in [6], they use $\nabla F(\mathbf{y}) \circ (1 - \mathbf{y})$. Both of them refer to the same thing, and are connected by Observation 2.2 in the paper [1].

The main idea of the improved algorithm is to impose a new constraint that $||z||_\infty \leq \alpha$ for some $\alpha > 0$, and at each time step $t \in [0,1]$, we instead solve: $z' = \text{argmax}_{z \in P, ||z||_\infty \leq \alpha} \big(\langle z, w(\mathbf{y}) \rangle\big)$ as the direction we change $\mathbf{y}$.

For the Measured Continuous Greedy algorithm [1], at each time step t, the value of the multilinear extension at the current position $\mathbf{y}(t)$, i.e. $F(\mathbf{y}(t))$, is increased by an amount proportional to $(1-||\mathbf{y}||_\infty)F(OPT)$. We want the amount to be as large as possible. What the new constraint $||z||_\infty \leq \alpha$ does is to slow down the growth of $||\mathbf{y}||_\infty$ at each time step, which leads to some improvement in $(1-||\mathbf{y}||_\infty)F(OPT)$.

Next, we briefly explain how does this new constraint bring an improvement to $(1-||\mathbf{y}(t)||_\infty)F(OPT)$ at each time step t.

We define the marginal gain of $F(\mathbf{y}(t))$ as: $\langle \mathbf{v}, w(\mathbf{y})\rangle$ where $\mathbf{v} = \text{argmax}_{z\in P}\big(\langle z, w(\mathbf{y})\rangle\big)$ and $w(\mathbf{y}) \in \mathbb{R}^U$, $(w(\mathbf{y}))_i = F(\mathbf{y} \vee \mathbf{1}_i) - F(\mathbf{y})$ for each $i \in U$.

Suppose at each time step t, we take into account the new constraint $||z||_\infty \leq \alpha$ and solve the following instead

$$z' = \text{argmax}_{z\in P, ||z||_\infty \leq \alpha}\langle z, w(\mathbf{y}(t))\rangle$$

If, at some time step, solving the above expression gives us a z' that is close to the optimal z^*. Then we can pick the moving direction z to at time step t to be a mixture of z' and z^*. By doing so, we have a direction that is close to the optimal z^* but with $||z||_\infty < ||z^*||_\infty$, hence, the marginal gain of $F(\mathbf{y}(t))$ at each time step t does not decrease by much, but at the same time, we can increase the gain: $(1-||\mathbf{y}(t)||_\infty)F(OPT)$ at t.

However, we might encounter the case where, we couldn't find a z' that is close to the optimal z^*. Then [6] defines a term called *well correlated*, and claim that the new direction z must be *well correlated* with the optimal z^*, and the implication is that we can therefore use some local search to find the new direction z, which is searching for $z = \text{argmax}_{\mathbf{x}\leq z'}F(\mathbf{x})$ to find a good fraction of the optimal z^* as our new moving direction z. (Note that for two vector $\mathbf{x}$ and $\mathbf{y}$, $\mathbf{x} \leq \mathbf{y}$ if $\mathbf{x}_i \leq \mathbf{y}_i$ for all $i \in U$). To identify a good fraction of the optimal z^*, [6] introduced an Double Greedy algorithm that is very similar to the algorithm of [3] for the USM problem, and it is trying to find $\max_{\mathbf{u}\leq\mathbf{x}\leq\mathbf{v}}F(\mathbf{x})$, under the box constraint $\{\mathbf{x} : \mathbf{u} \leq \mathbf{x} \leq \mathbf{v}\}$. The algorithm starts with the two initial solution $\mathbf{u}^{(0)} = \mathbf{u}$ and $\mathbf{v}^{(0)} = \mathbf{v}$, and then for each component $i \in U$, it computes the gradient signal at the x_i for both $\mathbf{u}$ and $\mathbf{v}$. And update $\mathbf{u}$ from left to right (i.e. increase), update $\mathbf{v}$ from right to left (i.e. decrease), then finally, $\mathbf{u}^{(i)}$ and $\mathbf{v}^{(i)}$ will converge to a same vector $\mathbf{u}^{(n)} = \mathbf{v}^{(n)}$ which equals to $\max_{\mathbf{u}\leq\mathbf{x}\leq\mathbf{v}}F(\mathbf{x})$.

The main algorithm of [6] consists of four stages:

Firstly, the algorithm runs a Continuous Greedy algorithm similar to [2] subject to the new constraint $||z||_\infty \leq \alpha$ and finds a solution of the updating direction $\mathbf{y}^{(\theta)}$, and this stage will continue within the time step interval $t \in [0, \theta]$, where $\theta \in [0, 1]$ will be updated in the main loop of the algorithm.

Secondly, the algorithm runs the standard Continuous Greedy Algorithm without the new constraint $||z||_\infty \leq \alpha$ and builds upon the solution from the first stage $\mathbf{y}^{(\theta)}$. This will take the rest of the time $t \in (\theta, 1]$.

Thirdly, the algorithm use the solution $\mathbf{y}^{(\theta)}$ from the first stage and first run an update without the new constraint, i.e. $\mathbf{p} = \text{argmax}_{z\in P}\langle z, w(\mathbf{y}^{(\theta)})\rangle$. Then run the Double Greedy algorithm to find a $\mathbf{z}$ that approximates $\text{argmax}_{\mathbf{x}\leq\mathbf{p}}F(\mathbf{x})$.

Finally, we pick the best solution among the solution from the first two stages (Continuous Greedy with new constraint plus without new constraint) and the solution from the third stage (Double Greedy Algorithm), based on who can achieve the highest F value.

Also, it is worth nothing that the outer most for loop select a threshold θ controlling how long we run the Continuous Greedy with the new constraint, and θ is increase gradually within $[0, 1]$. Within this main loop we keep running the Double Greedy Algorithm for the purpose that, once the solution from the previous two stages does not yield a promising increase in F, we switch to rely on the result from the Double Greedy, by previous description and the detailed analysis from section 5.2 in [6], a better solution must exist, since in this case, the solution from the first stage must be *well corrected* to the optimal direction z^*.

The algorithm efficiently constructs a solution $\mathbf{y} \in P$ such that $F(\mathbf{y}) \geq 0.372 \cdot F(OPT)$, where OPT is some optimal integer solution for the problem.

3.5.2 Further Improvement via a Non-symmetric Technique

A notable further improvement was given by Buchbinder and Feldman [8] which achieves the currently best approximation ratio of 0.385 for the non-monotone submodular function maximization problem subject to a matroid constraint.

Recall that in the algorithm of [6], during the Double Greedy stage, the technique of the unconstrained submodular maximization (USM) is used. [8] points out two inadequacies for the approach. Firstly, the idea used in the Measured Continuous Greedy is different from the idea used in USM, which makes it unnatural to combine the two different ideas together. Secondly, in the unconstrained case, the technique relies on a symmetry property of a non-negative submodular function f, that is, the complement of f, $f(U \setminus S)$ is also non-negative and submodular. In the constrained case, f no longer has such symmetry property, therefore, it seems even more unnatural to apply the technique borrowed from USM in this setting.

Instead of using the technique from USM, they use a local search method in their algorithm to build upon the contention resolution framework [5].

There are two main parts of the main algorithm. The first part is similar to the algorithm in [5] that is used to find a fractional solution to maximize the multilinear extension F.

The second part is called *Aided Measured Continuous Greedy* and the great point about this algorithm is that it use a very similar idea from the Measured Continuous Greedy algorithm [1] that it views the process of building up the solution as a process of moving a particle over the time interval [0,1]. This gets rid of the unnatural that the USM technique brings in to the previous algorithm [6].

The Aided Measured Continuous Greedy Algorithm takes in the output Z from the first part of the main algorithm as well as a time threshold t_s. Again, starting with an empty solution $\mathbf{y}(0)$ at time 0, the t_s divides the Aided Measured Continuous Greedy into two stages. During the first part of the time interval $t \in [0, t_s)$, the algorithm determines the update direction $\mathbf{x}(t)$ by solving following and grows the solution $\mathbf{y}(t)$ according to $\mathbf{x}(t)$.

$$\mathbf{x}(t) = \text{argmax}_{\mathbf{x} \in P} \Big[\sum_{e \in U \setminus Z} w_e(t) \cdot \mathbf{x}_e(t) - \sum_{e \in Z} \mathbf{x}_e(t) \Big]$$

i.e. the algorithm runs the Measured Continuous Greedy on the ground set $U \setminus Z$ and ignores all the elements in the output Z from part 1 to determine the update direction. (Note that $w_e(t)$ refers to the *residual increase* defined before).

Then for the second stage of the Aided Measure Continuous Greedy, it determines the update direction $\mathbf{x}(t)$ in the same way as in the Measured Continuous Greedy, i.e. solve: $\mathbf{x}(t) = \text{argmax}_{\mathbf{x} \in P} \big[\langle w(t), \mathbf{x}(t) \rangle \big]$ and grows $\mathbf{y}(t)$ accordingly. Finally, the output is $\mathbf{y}(1)$ at time 1, by the down-closed property of the matroid polytope P, $\mathbf{y}(1) \in P$.

Going back to the main algorithm, after obtain two solution s_1 from part 1, and s_2 from part two, the algorithm output a final solution s_1 with probability p and output s_1 with probability $1-p$. They determine the optimal p as well as the threshold time t_s, a non-convex program is set up (the objective function and some other details can be found in section 3 in [8]) and solves for the optimal $p = 0.230$ and $t_s = 0.372$.

The intuition for the main algorithm is that if the solution s_1 from part 1 is good enough, then s_1 is returned, else if s_1 is far from OPT in some sense, then s_1 provides some guidance for the main algorithm to improve the final result, specifically, to see how well the solution will improve if we exclude them from the ground set U.

The result of the main algorithm is that it can find a feasible solution $\mathbf{x}$ in polynomial time, such that, for a non-negative submodular function f subjecting to a matroid constraint, the multilinear extension F of f satisfies: $F(\mathbf{x}) \geq 0.385 \cdot f(OPT)$, where OPT is the optimal solution that maximizes f.

4 Summary

As we know, the problem of submodular function maximization subject to some constraints is NP-hard, how do we obtain a good approximation to this problem has been studied intensively over the years.

Comparing to using the traditional combinatorial or local search methods to approach this problem, the continuous framework [5] as well as the techniques presented in this section is rather novel. On one hand, they serve as the basis for the currently best approximation algorithm to the problem, on the other hand, the techniques involved in the development of new algorithm as well the corresponding analysis are relatively intuitive. For instance, viewing the process of building up the solution $F(\mathbf{y})$ to approaching the optimal $F(OPT)$ as a process of moving a particle along the time interval $[0, 1]$ is intuitive, under this framework, we can apply techniques from the theory of Differential Equation to this problem which has a discrete nature.

Equipped with an efficient way on rounding the continuous solution into some feasible integer solution, in general, the methods from continuous mathematics can certainly give us more guidance and provides insights for developing new approximation algorithm.

As mentioned in [6], for the case of monotone submodular function, the Continuous Greedy Algorithm provides the optimal approximation (the ratio is $1 - 1/e$). However, it still remains open on whether one could derive an efficient approximation algorithm that further improve the approximation ratio for the non-monotone case. The work in [6] takes a new approach and focus on analyzing the rate of change of $F(\mathbf{y})$ over time to further increase the value of $F(\mathbf{y})$ from the Continuous Greedy algorithm. And the work in [8] use a very different technique, namely a non-symmetric technique, which is more natural for the constrained submodular maximization problem that achieves the currently best approximation ratio of 0.385. The current hardness result is that it is impossible to find an algorithm that yields an approximation ratio better than 0.478 for maximizing a non-monotone submodular function subject to a matroid constraint, which is proven by Gharan and Vondrák [9].

A significant advantage of this two steps approach based on the multilinear extension of a submodular function is that it provides a general framework for the submodular function maximization problem and holds for an arbitrary matroid constraint. This provides a solution to the problem that traditional combinatorial methods bring in, which is the algorithms designed based on this highly rely on the particular structure of the problem setting. It is relatively hard to extend or generalize those algorithms.

It is worth mentioning that some drawback of this approach, although the algorithms based on this approach runs in polynomial time, they are often considered to be inefficient in practice. Numerous attempts have been made to obtain algorithms with faster running times. Among them, a notable approach is by parallelization and distributed algorithms, the relevant study is on the *adaptivity* complexity for constrained submodular function maximization starting from Balkanski and Singer [10]. The *adaptivity* of an algorithm is the number of sequential rounds of queries it makes to the value oracle of the function, in each round, the total number of oracle queries made in parallel should be polynomial. Essentially, there is a tradeoff between the approximation quality and adaptivity of the algorithm and [10] gave a randomized algorithm that achieves an approximation ratio of $1/3 - \epsilon$ using $O(\log n)$ rounds of sequential queries for maximizing a monotone submodular function subject to a cardinality constraint.

References

[1] M. Feldman, J. Naor, and R. Schwartz, "A Unified Continuous Greedy Algorithm for Submodular Maximization," *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science,* 2011.

[2] J. Vondrák, "Optimal approximation for the submodular welfare problem in the value oracle model," *Proceedings of the fourtieth annual ACM symposium on Theory of computing - STOC 08*, 2008.

[3] N. Buchbinder, M. Feldman, J. Seffi, and R. Schwartz, "A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization," *SIAM Journal on Computing*, vol. 44, no. 5, pp. 1384–1402, 2015.

[4] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a Submodular Set Function Subject to a Matroid Constraint (Extended Abstract)," *Integer Programming and Combinatorial Optimization Lecture Notes in Computer Science*, pp. 182–196, 2007.

[5] C. Chekuri, J. Vondrák, and R. Zenklusen, "Submodular Function Maximization via the Multilinear Relaxation and Contention Resolution Schemes," *SIAM Journal on Computing*, vol. 43, no. 6, pp. 1831–1879, 2014.

[6] A. Ene and H. L. Nguyen, "Constrained Submodular Maximization: Beyond 1/e," *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, 2016.

[7] A. Ageev and M. Sviridenko, "Pipage Rounding: A New Method of Constructing Algorithms with Proven Performance Guarantee," *Journal of Combinatorial Optimization*, vol. 8, no. 3, pp. 307–328, 2004.

[8] N. Buchbinder and M. Feldman, "Constrained submodular maximization via a non-symmetric technique," *arXiv preprint arXiv:1611.03253,* 2016.

[9] S. O. Gharan and J. Vondrák, "Submodular Maximization by Simulated Annealing," *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, 2011.

[10] E. Balkanski and Y. Singer, "The adaptive complexity of maximizing a submodular function," *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing - STOC*, 2018.