Fast Algorithms for Maximizing Monotone Nonsubmodular Functions

Bin Liu(✉) and Miaomiao Hu

School of Mathematical Sciences, Ocean University of China, Qingdao 266100, China
binliu@ouc.edu.cn

Abstract. Recently, there has been much progress on improving approximation for problems of maximizing monotone (nonsubmodular) objective functions, and many interesting techniques have been developed to solve these problems. In this paper, we develop approximation algorithms for maximizing a monotone function f with generic submodularity ratio γ subject to certain constraints. Our first result is a simple algorithm that gives a $(1 - e^{-\gamma} - \epsilon)$-approximation for a cardinality constraint using $O(\frac{n}{\epsilon} log \frac{n}{\epsilon})$ queries to the function value oracle. The second result is a new variant of the continuous greedy algorithm for a matroid constraint. We combine the variant of continuous greedy method and contention resolution schemes to find a solution with approximation ratio $(\gamma^2(1 - \frac{1}{e})^2 - O(\epsilon))$, and the algorithm makes $O(rn\epsilon^{-4} log^2 \frac{n}{\epsilon})$ queries to the function value oracle.

Keywords: Nonsubmodular function · Maximization · Cardinality constraint · Matroid constraint · Approximation algorithm

1 Introduction

In these years, optimization problems involving maximization of a set function have attracted much attention. Many combinatorial optimization problems can be formulated as the maximization of a set function. For example, the welfare maximization problem is a submodular function maximization problem. Although submodular functions have some good properties, such as diminishing marginal returns, and they also have important applications, many objective functions in practical problems are not submodular. In these settings, we turn to study the problem of maximizing nonsubmodular functions.

The problems of maximizing a submodular function subject to combinatorial constraints are generally NP-hard, so we turned to find approximation algorithms for solving these problems. The greedy approach is a basic technique for these problems: start from an empty set; iteratively add to the current solution

This work was supported in part by the National Natural Science Foundation of China (11971447, 11871442), and the Fundamental Research Funds for the Central Universities.

set one element that results in the largest marginal gain of the objective function while satisfying the constraints. Meanwhile, the continuous greedy approach is basic technique for maximizing a submodular function under a matroid constraint. We should note that a continuous greedy algorithm is always combined with some rounding methods in order to get the feasible solution, such as pipage rounding and swap rounding.

Given a ground set $N = \{1, 2, ..., n\}$, a set function f is *nonnegative* if $f(S) \geq 0$ for any $S \subseteq N$. The function f is *monotone* if $f(S) \leq f(T)$ whenever $S \subseteq T$. Moreover, f is called *submodular* if $f(S \cup \{j\}) - f(S) \geq f(T \cup \{j\}) - f(T)$ for any $S \subseteq T \subseteq N$, $j \in N \setminus T$. Without loss of generality, for any pair of $S, T \subseteq N$, denote by $f_S(T) = f(S \cup T) - f(S)$ the marginal gain of the set T in S. Specially, denote by $f_S(j) = f(S \cup \{j\}) - f(S)$ the marginal gain of the singleton set $\{j\}$ in S, for any $S \subseteq N$ and any $j \in N$. Moreover, we assume there is a value oracle for the function f.

In this paper, we deal with the following two optimization problems:

$$\textbf{Problem (1)}: \max\{f(S) : |S| \leq k, S \subseteq N\}$$
$$\textbf{Problem (2)}: \max\{f(S) : S \in \mathcal{I}, S \subseteq N\}$$

where $f : 2^N \rightarrow R_+$ is a monotone (nonsubmodular) function, k is a positive integer, and $(N, \mathcal{I})$ is a matroid.

In the previous studies, it is proved that, when f is nonnegative, monotone and submodular, the greedy approach yield a $(1 - \frac{1}{e})$-approximation for a cardinality constraint [14], which is also proved to be optimal [13]. After that, lots of results obtained for maximizing a submodular function subject to different constraints. But for nonsubmodular functions, there are only a few results. On the purpose of using known results or methods for maximizing submodular functions, one can define some parameters, such as submodularity ratio, to deal with the maximization of nonsubmodular functions. Das and Kempe [5] first defined the *submodularity ratio* $\hat{\gamma} = min_{S,T \subseteq N} \frac{\sum_{j \in T \setminus S} f_S(j)}{f_S(T)}$, which describes how close a function is to being submodular. Afterwards, Bian et al. [3] proved that the greedy approach gives a $\frac{1}{\alpha}(1 - e^{-\alpha\hat{\gamma}})$-approximation for maximizing a monotone nonsubmodular function with curvature α and submodularity ratio $\hat{\gamma}$ under a cardinality constraint. Recently, Nong et al. [15] proposed the *generic submodularity ratio* which is the largest scalar γ that satisfies $f_S(j) \geq \gamma f_T(j)$, for any $S \subseteq T \subseteq N$. What's more, they showed that the greedy algorithm can achieve a $(1 - e^{-\gamma})$-approximation and a $\frac{\gamma}{1+\gamma}$-approximation for maximizing a strictly monotone nonsubmodular function with generic submodularity ratio γ under a cardinality constraint and a matroid constraint respectively.

Continuous greedy algorithm is always combined with some rounding methods in solving the problem of maximizing a submodular function under a matroid constraint. In the previous studies, Vondrák [17] showed that there exists a $\frac{(1-e^{-\alpha})}{\alpha}$-approximation algorithm for any monotone submodular function with curvature α and matroid constraints, achieving by the continuous greedy approach and the pipage rounding technique [1]. Later, Badanidiyuru et al. [2] proposed an accelerated continuous greedy algorithm for maximizing a monotone submodular function under a matroid constraint using the multilinear extension and swap rounding, and they achieved a $(1-\frac{1}{e}-\epsilon)$-approximation. The pipage rounding and swap rounding technique are effective methods, but it depends on the submodularity of the objective function. In addition, Chekuri et al. [4] proposed the contention resolution schemes, another framework for rounding, and showed a $(1-\frac{1}{e})$-approximation for rounding a fractional solution under a matroid constraint. Recently, Gong et al. [9] combined the continuous greedy algorithm and contention resolution schemes technique to achieve a $\gamma(1-e^{-1})(1-e^{-\gamma}-O(1))$-approximation.

For nonsubmodular functions optimization, there are other research efforts in application-driving, such as supermodular-degree [7,8], difference of submodular functions [10,12,18], and discrete difference of convex functions [11,19].

Our Result. In this paper, our main contribution is to develop algorithms that have both theoretical approximation guarantees, and fewer queries of the function value oracle. We use simple decreasing threshold algorithm to solve the problem of maximizing a nonsubmodular function under a cardinality constraint. The following Theorem 1 implies the result in [2] for submodular functions (the case that $\gamma = 1$ in Theorem 1). Besides, we use the continuous greedy approach and contention resolution schemes to resolve the nonsubmodular maximizing problem under a matroid constraint. In Theorem 2, we improves the query times of a former result in [15], from $O(n^2)$ to $O(rn\epsilon^{-4}log^2\frac{n}{\epsilon})$. Formally, we obtain the following two theorems.

Theorem 1. *There is a $(1-e^{-\gamma}-\epsilon)$-approximation algorithm for maximizing a monotone function with generic submodularity ratio γ subject to a cardinality constraint, using $O(\frac{n}{\epsilon}log\frac{n}{\epsilon})$ queries to the function oracle.*

Theorem 2. *There is a $(\gamma^2(1-e^{-1})^2-O(\epsilon))$-approximation algorithm for maximizing a monotone function with generic submodularity ratio γ subject to a matroid constraint, using $O(rn\epsilon^{-4}log^2\frac{n}{\epsilon})$ queries to the function oracle.*

2 Preliminary

In this section, we propose some definitions and properties that we will use in the following of the paper.

Definition 1 (Generic Submodularity Ratio [15]).
Given a ground set N and a monotone set function $f: 2^N \rightarrow R_+$, the generic submodularity ratio of f is the largest scalar γ such that for any $S \subseteq T \subseteq N$ and any $j \in N \setminus T, f_S(j) \geq \gamma f_T(j)$.

Definition 2 (The Multilinear Extension [2]).
For a function $f: 2^N \rightarrow R_+$, we define the multilinear extension of f is $F(\mathbf{x}) = E[f(R(\mathbf{x}))]$, where $R(\mathbf{x})$ is a random set where element i appears independently with probability $\mathbf{x}$.

Definition 3 (Matroid [6]).
A matroid $\mathcal{M} = (N, \mathcal{I})$ can be defined as a finite N and a nonempty family $\mathcal{I} \subset 2^N$,called independent set, such that:
(i) $A \subset B, B \in \mathcal{I}$, then $A \in \mathcal{I}$;
(ii) $A, B \in \mathcal{I}, |A| \leq |B|$, then there is an element $j \in B \backslash A$, $A + j \in \mathcal{I}$.

Definition 4 (Matroid Polytope [9]).
Given a matroid $(N, \mathcal{I})$, the matroid polytope is defined as $\mathcal{P}_{\mathcal{I}} = conv\{1_I : I \in \mathcal{I}\} = \{\mathbf{x} \geq 0 : for\ any\ S \subset N; \sum_{j \in S} x_j \leq r_{\mathcal{I}}(S)$, where $r_{\mathcal{I}}(S) = max\{|I| : I \subset S, I \subset \mathcal{I}\}$ is the rank function of matroid $(N, \mathcal{I})$(hereinafter called r).

Definition 5 (Contention Resolution Schemes (CR Schemes) [4]).
For any $\mathbf{x} \in \mathcal{P}_{\mathcal{I}}$ and any subset $A \subset N$, a CR scheme π for $\mathcal{P}_{\mathcal{I}}$ is a removal procedure that returns a random set $\pi_{\mathbf{x}}(A)$ such that $\pi_{\mathbf{x}}(A) \subset A \cap support(\mathbf{x})$ where $support(\mathbf{x}) = \{j \in N | x_j > 0\}$ and $\pi_{\mathbf{x}}(A) \in (I)$ with probability 1.

Afterwards, Gong et al. [9] proved that the CR schemes have a $\gamma(1 - \frac{1}{e})$-approximation in the nonsubmodular setting, where γ is the generic submodularity ratio of the objective function.

Lemma 1 (Property of the generic submodularity ratio [9]).

(a) $\gamma \in [0, 1]$;
(b) f is submodular iff $\gamma = 1$;
(c) $\sum_{j \in T \backslash S} f_S(j) \geq \gamma f_S(T)$, for any $S, T \subseteq N$

Lemma 2 ([16]).
Let $\mathcal{M} = (N, \mathcal{I})$ be a matroid, and $B_1, B_2 \in \mathcal{B}$ be two bases. Then there is a bijection $\phi : B_1 \rightarrow B_2$ such that for every $b \in B_1$, we have $B_1 - b + \phi(b) \in \mathcal{B}$.

Lemma 3 ([2]). *(Relative + Additive Chernoff Bound) Let $X_1, X_2, ..., X_m$ be independent random variables such that for each i, $X_i \in [0, 1]$, and let $X = \frac{1}{m} \sum_{i=1}^{m} X_i$ and $\mu = E[X]$. Then*

$$Pr[X > (1 + \alpha)\mu + \beta] \leq e^{-\frac{m\alpha\beta}{3}},$$

and

$$Pr[X < (1 - \alpha)\mu - \beta] \leq e^{-\frac{m\alpha\beta}{2}}.$$

Note that the generic submodularity ratio of a strictly monotone function is greater than 0. In the following of the paper, we consider the problem of maximizing a nonnegative strictly monotone and normalized set function under certain constraints.

Algorithm 1. Simple Decreasing Threshold Algorithm

```
Input: $f : 2^N \to R_+, k \in \{1, 2, ..., n\}$.
Output: A set $S \subset N$ satisfying $|S| \le k$.
 1: $S \leftarrow \emptyset$;
 2: $d \leftarrow max_{j \in N} f(j)$;
 3: for ($w = \frac{d}{\gamma}; w \ge \frac{\epsilon d}{n\gamma}; w \leftarrow \frac{1-\epsilon}{\gamma} w$) do
 4:     for all $i \in N$ do
 5:         if $|S \cup \{j\}| \le k$ and $f_S(j) \ge w$ then
 6:             $S \leftarrow S \cup \{j\}$
 7:         end if
 8:     end for
 9: end for
10: return $S$
```

3 Cardinality Constraint

First we present a simple algorithm, Algorithm 1, for **Problem (1)**: $\max\{f(S) : |S| \le k, S \subseteq N\}$, where f is a monotone function with generic submodularity ratio γ. Our goal is to develop an algorithm that have both theoretical approximation guarantee, and fewer queries of function value oracle.

Next we prove Theorem 1. Firstly, we check the number of queries of Algorithm 1. Obviously, there are two loops in Algorithm 1. Each inner loop executes n queries of value oracle. According to the termination condition of the outer loop, we get the query numbers of per outer loop is $O(\frac{1}{\epsilon} log \frac{n}{\epsilon})$. Therefore the algorithm using $O(\frac{n}{\epsilon} log \frac{n}{\epsilon})$ queries of the value oracle. For the approximation ratio, it is necessary to prove the following claim.

Claim 1. Let O be an optimal solution. Given a current solution S at the beginning of each iteration, the gain of the element added to S is at least $\frac{1-\epsilon}{k} \sum_{a \in O \setminus S} f_S(a)$.

Proof. Suppose that the next element chosen is a and the current threshold value is w. Then it implies the following inequalities

$$\begin{cases} f_S(x) \ge w, & if\ x = a; \\ f_S(x) \le \frac{w}{1-\epsilon}, & if\ x \in O \setminus S \cup \{a\}. \end{cases}$$

The above inequalities imply that $f_S(a) \ge (1-\epsilon) f_S(x)$ for each $x \in O \setminus (S \cup \{a\})$. Taking an average over these inequations, we have

$$f_S(a) \ge \frac{1-\epsilon}{|O \setminus S|} \sum_{x \in O \setminus S} f_S(x) \ge \frac{1-\epsilon}{k} \sum_{x \in O \setminus S} f_S(x).$$

Now we finish the proof of Claim 1. □

Then it is straightforward to finish the proof of Theorem 1.

Proof. Consider on a solution $S_i = \{a_1, a_2, ..., a_i\}$. After i steps, by Claim 1, we have

$$f_{S_i}(a_{i+1}) \geq \frac{1-\epsilon}{k} \sum_{a \in O \setminus S_i} f_{S_i}(a).$$

Then

$$\sum_{a \in O \setminus S_i} f_{S_i}(a) \geq \gamma f_{S_i}(O) \geq \gamma(f(O) - f(S_i)).$$

Therefore,

$$f(S_{i+1}) - f(S_i) = f_{S_i}(a+1) \geq \frac{1-\epsilon}{k}\gamma(f(O) - f(S_i)).$$

We have

$$\begin{aligned} f(S_k) &\geq (1 - (1 - \frac{(1-\epsilon)\gamma}{k})^k) f(O) \\ &\geq (1 - e^{-\gamma(1-\epsilon)}) f(O) \\ &\geq (1 - e^{-\gamma} - \epsilon) f(O). \end{aligned}$$

So we complete the proof of Theorem 1. $\square$

4 Matroid Constraint

In this section we give a $(\gamma^2(1-\frac{1}{e})^2 - O(\epsilon))$-approximation algorithm for **Problem (2)**: $\max\{f(S) : S \in \mathcal{I}, S \subseteq N\}$, where f is a monotone function with generic submodularity ratio γ, using $O(rn\epsilon^{-4}log^2\frac{n}{\epsilon})$ queries to the value oracle. The general outline of our algorithm follows from the continuous greedy algorithm in [2]. With a fractional solution being built up gradually from $\mathbf{x} = \mathbf{0}$, and finally using the contention resolution schemes from [4] to convert the fractional solution to an integer one.

Notation. In the following, for $\mathbf{x} \in [0,1]^N$, we denote by $R(\mathbf{x})$ a random set that contains each element $i \in N$ independently with probability x_i. We denote $R(\mathbf{x} + \epsilon \mathbf{1}_S)$ as $R(\mathbf{x}, S)$. Before we analyse the approximation of Algorithm 2, we give and analyse a subroutine, which is used in Algorithm 2. This subroutine takes a current fractional solution $\mathbf{x}$ and adds to it an increment corresponding to an independent set B, to obtain $\mathbf{x} + \epsilon \mathbf{1}_B$. The way we find B in Algorithm 3 is similar to that in Algorithm 1.

Claim 2. Let O be an optimal solution. Given a fractional solution $\mathbf{x}$, Algorithm 3 produces a new fractional solution $\mathbf{x}' = \mathbf{x} + \epsilon \mathbf{1}_B$ such that

$$F(\mathbf{x}') - F(\mathbf{x}) \geq \epsilon(\gamma(1-\epsilon) - \frac{2\epsilon}{\gamma}) f(O) - F(\mathbf{x}').$$

Algorithm 2. Revised Continuous Greedy Algorithm

```
Input: $f : 2^N \to R_+,\ \mathcal{I} \subseteq 2^N$.
Output: A set $S \subseteq N$ satisfying $S \in \mathcal{I}$.
 1: $\mathbf{x} \leftarrow \mathbf{0}$;
 2: for $(t \leftarrow \epsilon; t \leq 1; t \leftarrow t + \epsilon)$ do
 3:     $B \leftarrow$ the output of Algorithm 3
 4:     $\mathbf{x} \leftarrow \mathbf{x} + \epsilon \mathbf{1}_B$
 5: end for
 6: $S \leftarrow$ contention resolution schemes $(\mathbf{x}, \mathcal{I})$
 7: return $S$
```

Algorithm 3. Decreasing Threshold procedure

```
Input: $f : 2^N \to R_+,\ \mathbf{x} \in [0,1]^N,\ \epsilon \in [0,1],\ \mathcal{I} \subseteq 2^N$.
Output: A set $B \subset N$ satisfying $B \in \mathcal{I}$
 1: $B \leftarrow \emptyset$;
 2: $d \leftarrow max_{j \in N} f(j)$;
 3: for $(w = \frac{d}{\gamma}; w \geq \frac{\epsilon d}{r\gamma}; w \leftarrow \frac{1-\epsilon}{\gamma} w)$ do
 4:     for all $e \in N$ do
 5:         $w_e(B, \mathbf{x}) \leftarrow$ estimate of $E[f_{R(\mathbf{x}+\epsilon\mathbf{1}_B)}(e)]$ by averaging $\frac{rlogn}{\epsilon^2}$ random samples.
 6:         if $B \cup \{e\} \in \mathcal{I}$ and $w_e(B, \mathbf{x}) \geq w$ then
 7:             $B \leftarrow B \cup \{e\}$
 8:         end if
 9:     end for
10: end for
11: return $B$
```

Proof. Suppose that Algorithm 3 returns r elements, $B = \{b_1, b_2, ..., b_r\}$(indexed in the order in which they were chosen). In fact, Algorithm 3 might return fewer than r elements if the threshold w drops below $\frac{\epsilon d}{r}$ before termination. In this case, we formally add dummy elements of value 0 so that $|B| = r$.

Let $O = \{o_1, o_2, ..., o_r\}$ be an optimal solution, with $\phi(b_i) = o_i$ as specified by Lemma 2. Additionally, let B_i denote the first i elements of B, and let O_i denote the first i elements of O.

Note that by Lemma 3, we get that there is an error while using $w_e(B_i, \mathbf{x})$ to estimate $E[f_{R(\mathbf{x},B_i)}(e)]$, with high probability we have the following inequality

$$|w_e(B_i, \mathbf{x}) - E[f_{R(\mathbf{x},B_i)}(e)]| \leq \frac{\epsilon f(O)}{\gamma r} + \epsilon E[f_{R(\mathbf{x},B_i)}(e)]. \tag{1}$$

When an element b_i is chosen, o_i is a candidate element which could have been chosen instead of b_i. Thus, according to Algorithm 3, and because either o_i is a potential candidate of value within a factor of $1 - \epsilon$ of the element we chose instead, or the algorithm terminated and all remaining elements have value below $\frac{\epsilon d}{\gamma r}$, we have

$$w_{b_i}(B_{i-1}, \mathbf{x}) \geq (1 - \epsilon) w_{o_i}(B_{i-1}, \mathbf{x}) - \frac{\epsilon d}{\gamma r}. \tag{2}$$

Combining (1) and (2), and the fact that $f(O) \geq d$, we have

$$E[f_{R(\mathbf{x},B_{i-1})}(b_i)] \geq (1-\epsilon)E[f_{R(\mathbf{x},B_{i-1})}(o_i)] - 2\frac{\epsilon f(O)}{\gamma r}. \tag{3}$$

Then at each step in Algorithm 2:

$$\begin{aligned}
F(\mathbf{x}') - F(\mathbf{x}) &= F(\mathbf{x} + \epsilon \mathbf{1}_B) - F(\mathbf{x}) \\
&= \sum_{i=1}^{r} (F(\mathbf{x} + \epsilon \mathbf{1}_{B_i}) - F(\mathbf{x} + \epsilon \mathbf{1}_{B_{i-1}})) \\
&= \sum_{i=1}^{r} \epsilon \frac{\partial F}{\partial x_{b_i}}\Big|_{\mathbf{x}+\epsilon \mathbf{1}_{B_{i-1}}} \\
&\geq \sum_{i=1}^{r} \epsilon E[f_{\mathbf{x}+\epsilon \mathbf{1}_{B_{i-1}}})(b_i)] \\
&\geq \sum_{i=1}^{r} \epsilon((1-\epsilon)E[f_{\mathbf{x}+\epsilon \mathbf{1}_{B_{i-1}}})(o_i)] - 2\frac{\epsilon f(O)}{\gamma r}) \\
&\geq \sum_{i=1}^{r} \epsilon((1-\epsilon)\gamma E[f_{R(\mathbf{x}+\epsilon \mathbf{1}_{B \cup \{o_1, o_2, \ldots, o_{i-1}\}})}(o_i)] - 2\frac{\epsilon f(O)}{\gamma r}) \\
&= \epsilon((1-\epsilon)\gamma E[f(R(\mathbf{x}') \cup O) - f(R(\mathbf{x}'))] - 2\frac{\epsilon f(O)}{\gamma} \\
&\geq (\gamma\epsilon(1-\epsilon) - \frac{2\epsilon^2}{\gamma})f(O) - \epsilon F(\mathbf{x}') \\
&= \epsilon((\gamma(1-\epsilon) - \frac{2\epsilon}{\gamma})f(O) - F(\mathbf{x}')).
\end{aligned}$$

The second inequality follows from that o_i is a candidate element when b_i is chosen. The first and last inequalities are due to monotonicity, and the third inequality is due to the definition of generic submodularity ratio γ. □

Claim 3. Algorithm 3 makes $O(\frac{1}{\epsilon^3} nrlog^2 \frac{n}{\epsilon})$ queries to the function oracle.

Proof. Obviously, there are two loops in Algorithm 3. According to the termination condition of the outer loop, we get the query numbers of per outer loop is $O(\frac{1}{\epsilon} log \frac{n}{\epsilon})$. The number of iterations in the inner loop is n, and the number of samples per evaluation of F is $\frac{1}{\epsilon^2} rlogn$ in per inner loop. Therefore, Algorithm 3 makes $O(\frac{1}{\epsilon^3} nrlog^2 \frac{n}{\epsilon})$ queries to the value oracle. □

Claim 4. Algorithm 2 has an approximation ratio of $\gamma^2(1-\frac{1}{e})^2 - O(\epsilon)$.

Proof. Define $\Omega = (\gamma(1-\epsilon) - \frac{2\epsilon}{\gamma})f(O)$. Substituting this in the result of Claim 2, we have

$$F(\mathbf{x}(t+\epsilon)) - F(\mathbf{x}(t)) \geq \epsilon(\Omega - F(\mathbf{x}(t+\epsilon))).$$

Add $\Omega - F(\mathbf{x}(t+\epsilon))$ to the inequation we have

$$\Omega - F(\mathbf{x}(t+\epsilon)) \leq \frac{\Omega - F(\mathbf{x}(t))}{1+\epsilon}.$$

Using induction to this inequation, we have

$$\Omega - F(\mathbf{x}(t)) \leq \frac{\Omega}{(1+\epsilon)^{\frac{t}{\epsilon}}}.$$

Substituting $t = 1$ and rewriting the inequation, we have

$$\begin{aligned} F(\mathbf{x}(t)) &\geq (1 - \frac{1}{(1+\epsilon)^{\frac{1}{\epsilon}}})\Omega \\ &= (1 - \frac{1}{(1+\epsilon)^{\frac{1}{\epsilon}}})(\gamma(1-\epsilon) - \frac{2\epsilon}{\gamma}) \\ &\geq \gamma(1 - \frac{1}{e}) - O(\epsilon). \end{aligned}$$

Besides, when we use CR schemes to convert the fractional solution to the integer one, there also have an approximation ratio which is $\gamma(1-\frac{1}{e})$.

Therefore, the approximation ratio of Algorithm 2 is $\gamma^2(1-\frac{1}{e})^2 - O(\epsilon)$. □

Claim 5. Algorithm 2 makes $O(\frac{1}{\epsilon^4}nrlog^2\frac{n}{\epsilon})$ queries to the function oracle.

Proof. Observe that in Algorithm 2, the queries to the function oracle is only related to Algorithm 3. Therefore the total number of oracle calls to the function is equal to the number of the loop multiplied with the number of oracle calls in one iteration. So we get the queries to the function oracle are at most $O(\frac{1}{\epsilon^4}nrlog^2\frac{n}{\epsilon})$. □

References

1. Ageev, A.A., Sviridenko, M.I.: Pipage rounding: a new method of constructing algorithms with proven performance guarantee. J. Comb. Optimiz. **8**(3), 307–328 (2014). https://doi.org/10.1023/B:JOCO.0000038913.96607.c2
2. Badanidiyuru, A., Vondrák, J.: Fast algorithm for maximizing submodular functions. In: SODA, pp. 1497–1514 (2014)
3. Bian, A.A., Buhmann, J.M., Krause, A., Tschiatschek, S.: Guarantees for greedy maximization of nonsubmodular functions with applications. In: ICML (2017)
4. Chekuri, C., Vondrk, J., Zenklusen, R.: Submodular function maximization via the multilinear relaxation and contention resolution schemes. SIAM J. Comput. **43**(6), 1831–1879 (2014)
5. Das, A., David, K.: Submodular meets spectral: greedy algorithms for subset selection, sparse approximation and dictionary selection. In: ICML (2011)
6. Edmonds, J.: Submodular functions, matroids, and certain polyhedra. In: Jünger, M., Reinelt, G., Rinaldi, G. (eds.) Combinatorial Optimization—Eureka, You Shrink!. LNCS, vol. 2570, pp. 11–26. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36478-1_2

7. Feldman, M., Izsak, R.: Constrained monotone function maximization and the supermodular degree. In: International Workshop and International Workshop on Approximation Randomization and Combinatorial Optimization Algorithms and Techniques, pp. 160–175 (2014)
8. Feige, U., Izsak, R.: Welfare maximization and the supermodular degree. In: Proceedings of the 4th Conference on Innovations in Theoretical Computer Science, pp. 247–256. ACM (2013)
9. Gong, S., Nong, Q., Liu, W., et al.: Parametric monotone function maximization with matroid constraints. J. Global Optimiz. **75**(3), 833–849 (2019). https://doi.org/10.1007/s10898-019-00800-2
10. Iyer, R., Bilmes, J.: Algorithms for approximate minimization of the difference between submodular functions, with applications. In: UAI, pp. 407–417 (2012)
11. Maehara, T., Murota, K.: A framework of discrete DC programming by discrete convex analysis. Math. Program. **152**(1–2), 435–466 (2015). https://doi.org/10.1007/s10107-014-0792-y
12. Narasimhan, M., Bilmes, J.A.: A submodular-supermodular procedure with applications to discriminative structure learning. arXiv:1207.1404 (2012)
13. Nemhauser, G.L., Wolsey, L.A.: Best algorithms for approximating the maximum of a submodular set functions. Math. Oper. Res. **3**(3), 177–188 (1978)
14. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions-I. Math. Program. **14**(1), 265–294 (1978). https://doi.org/10.1007/BF01588971
15. Nong, Q., Sun, T., Gong, S., et al.: Maximize a monotone function with a generic submodularity ratio. In: AAIM (2019)
16. Sviridenko, M.: A note on maximizing a submodular set function subject to a Knapsack constraint. Oper. Res. Lett. **32**(1), 41–43 (2004)
17. Vondrák, J.: Submodularity and Curvature: The Optimal Algorithm. Rims Kokyuroku Bessatsu B (2010)
18. Wu, W.-L., Zhang, Z., Du, D.-Z.: Set function optimization. J. Oper. Res. Soc. China **7**(2), 183–193 (2018). https://doi.org/10.1007/s40305-018-0233-3
19. Wu, C., et al.: Solving the degree-concentrated fault-tolerant spanning subgraph problem by DC programming. Math. Program. **169**(1), 255–275 (2018). https://doi.org/10.1007/s10107-018-1242-z