Approximation Algorithms for α-bisubmodular Function Maximization subject to matroid constraint

1

Original Research Article

Abstract

We design an approximation algorithm for maximizing α-bisubmodular function with matroid constraint, where the α-bisubmodular function is a generalization of a bisubmodular function. The concept of α-bisubmodularity is provided by Huber, Krokhin, and Powell[(7), 2014], rank function of delta-matroids and the cut capacity of directed networks have α-bisubmodularity. We consider the two cases of the problem, monotone and non-monotone objective function, respectively. We also show that the running time is polynomial.

Keywords: α-Bisubmodular function; Combinatorial optimization; Greedy algorithm; Matroid constraint

2010 Mathematics Subject Classification: 53C25; 83C05; 57N16

1 Introduction

Let E denote a finite nonempty set with size n and $3^E := \{(X_1, X_2)|X_1, X_2 \subseteq E, X_1 \cap X_2 = \emptyset\}$. A function $f : 3^E \to \mathbb{R}$ is called α-bisubmodular if for any $\mathrm{x} = (X_1, X_2)$ and $\mathrm{y} = (Y_1, Y_2)$ in 3^E,

$$f(\mathrm{x}) + f(\mathrm{y}) \geq f(\mathrm{x} \sqcap \mathrm{y}) + \alpha f(\mathrm{x} \sqcup \mathrm{y}) + (1 - \alpha) f(\mathrm{x} \dot\sqcup \mathrm{y})$$

where

$$\mathrm{x} \sqcup \mathrm{y} = (X_1 \cup Y_1 \backslash (X_2 \cup Y_2), X_2 \cup Y_2 \backslash (X_1 \cup Y_1))$$
$$\mathrm{x} \sqcap \mathrm{y} = (X_1 \cap X_2, Y_1 \cap Y_2)$$
$$\mathrm{x} \dot\sqcup \mathrm{y} = (X_1 \cup Y_1, X_2 \cup Y_2 \backslash (X_1 \cup Y_1))$$

f is bisubmodular function iff $\alpha = 1$.

Submodular function has been studied for decades and there is a beautiful line of research in this area, we just name a few here. Fisher, Nemhauser and Wolsey presented a sequence of papers [(10; 11; 4), 1978], showing that maximizing a monotone submodular function under cardinality constraint by greedy method could achieve $1 - 1/e$ approximation ratio to the optimum. Feige [(3), 1998] showed the hardness of approximation is $1 - 1/e$. Later, inspired by these results, Sviridenko in [(16), 2004] designed a simple algorithm based on greedy method also achieving $1 - 1/e$ ratio for knapsack constraint. As a natural generalization for cardinality constraint, matroid constraint was considered by Calinescu et al. [(2), 2011] via continuous greedy and multilinear extension, also achieving the same ratio as the cardinality constraint's.

In recent years, k-submodular has draw much attention and been widely covered. The extension of submodular function is k-submodular function in that any input is required to partition into k subsets. For unconstraint case, Ward and Živný[(18), 2016] first gave a deterministic greedy algorithm with approximation ratio of $\frac{1}{3}$, and a randomized version with $\frac{1}{1+\max\{1,\sqrt{\frac{k-1}{4}}\}}$. Iwata et al. [(9), 2016] followed the framework of randomized algorithm in [(18)], achieving $1/2$ approximation ratio which is tight for large k. If f is restricted to monotone, Ward and Živný [(18), 2016] provided a $1/2$ approximation algorithm. Iwata et al.[(9), 2016] further proved that there exists a $\frac{k}{2k-1}$-approximation algorithm. Oshima [(13), 2021] used the algorithm from [(9), 2016] with different probability distribution, presented a $\frac{k^2+1}{2k^2+1}$-approximation algorithm with sophisticated analysis. Many results for nonnegative k-submodular maximization problem with constraints were also covered. Ohsaka and Yoshida [(12), 2015] considered the total size constraint and individual size constraint, presenting $1/2$ and $1/3$-approximation algorithm, respectively. Sakaue [(14), 2017] presented a $1/2$-approximation algorithm with a matroid constraint. Tang et al. [(17), 2022] designed a combinatorial algorithm for maximizing monotone k-submodular with a knapsack constraint and its approximation ratio is $\frac{1-1/e}{2}$. Huber et al. [(7), 2014] first provided α-bisubmodular as a generalization of bisubmodular function, where bisubmodular is 2-submodular. α-bisubmodular was called skew bisubmodualr in [(7), 2014], they investigated the structures of α-bisubmodular. Fujishige et al. [(5), 2014]demonstrated the relationship between the skew bisubmodularity and a convex extension over rectangles. Fujishige and Tanigawa [(6), 2018] gave a polynomial combinatorial algorithms for α-bisubmodular function minimization. For the maximization of α-bisubmodular problem, Iwata et al. [(9), 2016] extended their algorithm to α-bisubmodular, providing a solution within a factor of $\frac{2\sqrt{\alpha}}{(1+\sqrt{\alpha})^2}$. Combining this result with another simple algorithm based on the structure of bisubmodular, the ratio can be improved to $\frac{8}{25}$ for any $\alpha \in [0, 1]$. Iwata et al. [(8), 2013] studied the inapproximability result for the α-bisubmodular function maximization, they derived that any algorithm which can return a more than 0.5 approximation ratio solution would requires an exponential number of queries in the value oracle model.. Shi et al.[(15), 2021] studied maximization of α-bisubmodular function subject to individual constraint, by using decreasing threshold method from [(1), 2013], achieving $1 - 1/e - \epsilon$-approximation with running $O(\frac{n}{\epsilon} \log \frac{n}{\epsilon})$ time.

In this paper,we extended the method from [(14), 2017] to devise an approximation algorithm for maximizing non-negative α-bisubmodular function with matroid constraint. Matroid constraint is a generalization of the individual constraint from [(15), 2021]. Matroid is a pair of $(E, \mathcal{I})$, where $\mathcal{I} \subseteq 2^E$ and is called the family of independent sets such that the following condition holds:

1. $\emptyset \in \mathcal{I}$
2. If $A \subseteq B \in \mathcal{I}$ then $A \in \mathcal{I}$
3. If $A, B \in \mathcal{I}$ and $|A| < |B|$, then there exists element $e \in B \backslash A$ such that $A \cup \{e\} \in \mathcal{I}$

We call a set *A independent* is A belongs $\mathcal{I}$ and *maximal* if no $B \in \mathcal{I}$ satisfies $A \nsubseteq B$. The set of all maximal sets in $\mathcal{I}$ is denoted by $\mathcal{B}$. Let r be the number of elements of a maximal set. By the third condition of matroid definition, it easily see that all maximal sets have the same number of elements. Many structures can be viewed as special cases of matroid, we name a few here:

1. Suppose that E is a finite set and $\mathcal{I} := \{I \subseteq E \| I| \leq k\}$, where k is a nonnegative integer.
2. E is the set of columns of a matrix and $\mathcal{I}$ is family set of linearly independent columns of E.

3. Let E be the set of edges of an undirected graph G, and $\mathcal{I}$ is the family set of forests of G.

2 Preliminaries

Given any two elements $\mathrm{x} = (X_1, X_2)$ and $\mathrm{y} = (Y_1, Y_2)$ in 3^E, an empty set is defined $0 = (\emptyset, \emptyset)$. We define $\mathrm{supp}(\mathrm{x}) := \{e \in E | e \in X_1 \cup X_2\}$. If $e \in X_i$, we write $\mathrm{x}(e) = i$, and if e is not contained in X_1 and X_2, then $\mathrm{x}(e) = 0$, i is called the position of e. We use $\mathrm{x} \sqcup (e, i)$ to represent the addition of e to X_i if $e \notin X_1 \cup X_2$. Suppose that a partial order $\preceq$ is defined on 3^E and $\mathrm{x} \preceq \mathrm{y}$ iff $X_1 \subseteq Y_1$ and $X_2 \subseteq Y_2$.

The marignal gain is the value of the addition of e to the i-th set of x, that is,

$$\Delta_{e,i} f(x) := f(X_i \sqcup (e, i), X_j) - f(X_i, X_j)$$

where $i,\ j \in [2]$ and $i \neq j$. It is not hard to see that α-submodular function f possesses *orthant submodularity*, that is,

$$\Delta_{e,i} f(\mathrm{x}) \geq \Delta_{e,i} f(\mathrm{y}), \text{ for any } \mathrm{x},\ \mathrm{y} \in 3^E \text{ with } \mathrm{x} \preceq \mathrm{y},\ e \notin Y_1 \cup Y_2$$

We say f has monotonicity if and only if $f(\mathrm{x}) \leq f(\mathrm{y})$ for any $\mathrm{x} \preceq \mathrm{y}$.

By the above properties, we can deduce the *α-pairwise monotonicity* of α-bisubmodular:

Theorem 2.1. If f is a α-bisubmodular function, *α-pairwise monotonicity* is $\alpha\Delta_{e,1} f(\mathrm{x}) + \Delta_{e,2} f(\mathrm{x}) \geq 0$, for any $e \notin \mathrm{supp}(\mathrm{x})$.

Proof. According to the definition, we have $f(X_1 \cup \{e\}, X_2) + f(X_1, X_2 \cup \{e\}) \geq f(X_1, X_2) + \alpha f(X_1, X_2) + (1 - \alpha) f(X_1 \cup \{e\}, X_2)$, which is identical to $\alpha\Delta_{e,1} f(\mathrm{x}) + \Delta_{e,2} f(\mathrm{x}) \geq 0$. □

An instance of maximizing α-bisubmodular subject to matroid constraint is described as follows. Given an α-bisumodular function $f : 2^E \to \mathbb{R}_+$, and a matroid $(E, \mathcal{I})$, solve

$$\begin{aligned} \max \quad & f(S) \\ \text{s.t.} \quad & S \in \mathcal{I} \end{aligned} \tag{2.1}$$

Before introducing the algorithm, we propose two important lemmas which will be used later.

Lemma 2.2. Let $A \in \mathcal{I}$ and $B \in \mathcal{B}$ such that $A \subsetneq B$. Then, for any $e \notin A$ satisfying $A \cup \{e\} \in \mathcal{I}$, there exists $e' \in B \backslash A$ such that $B \backslash \{e'\} \cup \{e\} \in \mathcal{I}$.

Proof. Let $S = (A \cap B) \cup \{e\}$. Since $A \cup \{e\} \in \mathcal{I}$, from the second condition of the matroid definition, we have $S \in \mathcal{I}$. $|S| < r$ holds because of $A \subsetneq B$. By the third condition from the matroid definition, we can add elements from B to the set S until $|S| = r$ and $S \in \mathcal{I}$. Then S consists of e and $r - 1$ elements from B, following the truth that $B \in \mathcal{B}$. Thus we have $S = B \backslash \{e'\} \cup \{e\} \in \mathcal{I}$. Finally, let e' be the element of $B \backslash S$, then we can obtain $e' \in B \backslash A$ since $A \cap B \in S$. □

Lemma 2.3. Any maximal solution for problem (2.1) has size r.

Proof. Let o be a maximal optimal solution with $|\mathrm{supp}(o)| < r$. Suppose that x is an element of 3^E satisfying that $\mathrm{supp}(\mathrm{x}) = r$. By the third condition of definition of matroid, an element $e \in \mathrm{supp}(\mathrm{x}) \backslash \mathrm{supp}(\mathrm{o})$ must exist and have $\mathrm{supp}(\mathrm{o}) \cup \{e\} \in \mathcal{I}$. Since f is α-bisubmodular, we have

$$\alpha\Delta_{e,1} f(\mathrm{o}) + \Delta_{e,2} f(\mathrm{o}) \geq 0$$

It implies that both $\Delta_{e,1} f(\mathrm{o})$ and $\Delta_{e,2} f(\mathrm{o})$ equal to 0, since o is an optimal solution. Then we can add e to arbitrary $i \in [2]$, this operation does not change the optimal value. Hence we can add some elements to $\mathrm{supp}(\mathrm{o})$ until its cardnality is r. □

We consider the two cases of problem (2.1), Case 1: f is monotone and Case 2: f is non-monotone. Both cases use the $GREEDY$ algorithm. The algorithm is described as follows. In the first step, algorithm initialize an zero vector s. In step 2, the algorithm requires r. If $|\text{supp}(\mathrm{s})| < r$, there exists at least one element can be added to supp(s), since the α-pariwise monotonicity of f and the third condition of the definition of matroid. At step 3, let $\mathcal{I}(s)$ denote all the element $e \in E \backslash \text{supp}(\mathrm{s})$ such that $\text{supp}(\mathrm{s}) \cup \{e\} \in \mathcal{B}$. At step 4, the marginal gain $\Delta_{e,i} f(\mathrm{s})$ is computed for all elements in $\mathcal{I}(\mathrm{s})$ and $i \in \{1, 2\}$. In the next step, the element e will be added to supp(s) and assigned the label i corresponding to the maximal margin gain. At the end of algorithm, it easily see that the final vector would satisfy the matroid constraint. We also show that our algorithm incurs $O(rn(IO + EO))$ computation cost, where IO and EO represent the time for independence oracle of the matroid and the evaluation oracle of the α-bisubmodular function, respectively.

The $GREEDY$ algorithm:

step 1: $\mathrm{s} \leftarrow 0$
step 2: for $j = 1$ to r do
step 3: Construct $\mathcal{I}(\mathrm{s})$ using the independence oracle
step 4: $\Delta_{e,i} f(\mathrm{s}) \leftarrow \max_{e \in E \backslash \text{supp}(\mathrm{s})} \{\Delta_{e,1} f(\mathrm{s}), \Delta_{e,2} f(\mathrm{s})\}$
step 6: $\mathrm{s}(e) \leftarrow i$
step 7: Endfor
step 8: Return s

3 Monotone case

In this section, let f be monotone, then we will prove that the $GREEDY$ algorithm returns a 0.5-approximate solution and ends in polynomial time. The following is the main theorem:

Theorem 3.1. Let $f : 2^E \to \mathbb{R}_+$ be a monotone α-bisubmodular function. The $GREEDY$ algorithm is a 0.5-approximation algorithm for problem (2.1) with running time $O(rn(IO + EO))$.

Proof. First we need to define some notations. Suppose that (e_j, i_j) is the pair chosen at the jth iteration, and s^j is the current solution after the jth iteration. Note that $\mathrm{s}^0 = 0$ and $\mathrm{s}^r = \mathrm{s}$. Let o be the optimal solution of the problem. In the following, we will construct a sequence of vectors $\mathrm{o}^0 = \mathrm{o}, \mathrm{o}^1, \ldots, \mathrm{o}^r$ such that $\mathrm{s}^j \preceq o^j$ and $\text{supp}(\mathrm{o}^j) \in \mathcal{B}$ for all $j \in [r]$. These notations will be used in the analysis of the algorithm.

Now we describe how to construct o^j from o^{j-1}. Assume that o^{j-1} has been constructed and satisfy the above property. In the j-th iteration, e_j is chosen into the current set $\text{supp}(\mathrm{s}^{j-1})$ and $\text{supp}(\mathrm{s}^{j-1}) \cup \{e_j\} \in \mathcal{I}$. Then there must exist an element $e' \in \text{supp}(\mathrm{o})$ such that $\text{supp}(\mathrm{o}) \backslash \{e'\} \cup \{e_j\} \in \mathcal{B}$ by the Lemma (2.2). Let $o_j = e'$, we define $\mathrm{o}^{j-\frac{1}{2}}$ by assigning 0 to the the position o_j of the o^{j-1}. And o^j is defined by assigning i_j to the position of e_j of $\mathrm{o}^{j-\frac{1}{2}}$. The vector thus constructed, and

$$\text{supp}(\mathrm{o}^j) = \text{supp}(\mathrm{o}^{j-1}) \backslash \{o_j\} \cup \{e_j\} \in \mathcal{B}$$

We can also verify that

$$\mathrm{s}^{j-1} \preceq \mathrm{o}^{j-\frac{1}{2}},$$

and o^j have the following property:

$$\mathrm{s}^j \preceq \mathrm{o}^j,$$

if $j = 0, 1, \ldots, r-1$, and $\mathrm{s}^r = \mathrm{o}^r = \mathrm{s}$. In the following, we give the analysis of the $GREEDY$ algorithm for maximizing monotone α-bisubmodular functions subject to matroid constraint.

The core of our proof is the following inequality for $j \in [r]$:

$$f(\mathrm{o}^{j-1}) - f(\mathrm{o}^j) \le f(\mathrm{s}^j) - f(\mathrm{s}^{j-1}) \tag{3.1}$$

Since $s^{j-1} \preceq o^{j-1}$ and $\mathrm{supp}(o^{j-1}) \in \mathcal{B}$, thus we have $\mathrm{supp}(s^{j-1}) \cup \{o_j\} \in \mathcal{B}$. It means o_j is a candidate member to be selected in the jth iteration of the *GREEDY* algorithm, so we have $\Delta_{o_j, o^{j-1}(o_j)} f(s^{j-1}) \leq \Delta_{e_j,i_j} f(s^{j-1})$ by the greedy rule.

Using the above property, we have

$$\begin{aligned} f(o^{j-1}) - f(o^j) &= f(o^{j-1}) - f(o^{j-\frac{1}{2}}) - [f(o^j) - f(o^{j-\frac{1}{2}})] \\ &= \Delta_{o_j, o^{j-1}(o_j)} f(o^{j-\frac{1}{2}}) - \Delta_{e_j,i_j} f(o^{j-\frac{1}{2}}) \\ &\leq \Delta_{o_j, o^{j-1}(o_j)} f(o^{j-\frac{1}{2}}) \\ &\leq \Delta_{o_j, o^{j-1}(o_j)} f(s^{j-1}) \\ &\leq \Delta_{e_j,i_j} f(s^{j-1}) \\ &= f(s^j) - f(s^{j-1}) \end{aligned}$$

where the first inequality holds since f is a monotone function, it implies that $\Delta_{e_j,i_j} f(o^{j-\frac{1}{2}}) \geq 0$. The second inequality is true from the orthant submodularity. Third inequality follows the greedy rule of the algorithm

By the formulation (3.1), we have

$$f(o) - f(s) = \sum_{j=1}^{r} (f(o^{j-1}) - f(o^j)) \leq \sum_{j=1}^{r} (f(s^j) - f(s^{j-1})) = f(s)$$

it is implies that $f(s) \geq \frac{f(o)}{2}$. Thus the approximation ratio for monotone case is proved, the only thing left is to show the running time. As we can see that the *for* loop runs r iterations. In each iteration, the algorithm access independence oracle at most n times in step 3, and evaluate f at most $2n$ times. Thus, *GREEDY* algorithm runs in $O(rn(IO + EO))$ time □

4 Non-monotone case

In this section, we show that *GREEDY* algorithm produces a $(2 + \frac{1}{\alpha})$-approximation solution for non-monotone case with the polynomial running time.

Theorem 4.1. Let $f : 2^E \to \mathbb{R}_+$ is a non-monotone α-bisubmodular function, then there exists a $(2 + \frac{1}{\alpha})$-approximation algorithm for problem (2.1) with running time $O(rn(IO + EO))$.

Proof. We give a new similar inequality of formulation (3.1) using for the proof of Theorem (4.1). For $j \in [r]$, we have

$$f(o^{j-1}) - f(o^j) \leq (1 + \frac{1}{\alpha})(f(s^j) - s^{j-1}) \tag{4.1}$$

To prove the inequality, we study two cases by the value of i_j. First we assume that $i_j = 1$, according to the α-pairwise monotonicity, we have

$$\alpha \Delta_{e_j,i_j} f(o^{j-\frac{1}{2}}) + \Delta_{e_j,2} f(o^{j-\frac{1}{2}}) \geq 0$$

thus we have $\Delta_{e_j,i_j} f(o^{j-\frac{1}{2}}) \geq -\frac{1}{\alpha} \Delta_{e_j,2} f(o^{j-\frac{1}{2}})$. When $i_j = 2$, we have

$$\Delta_{e_j,i_j} f(o^{j-\frac{1}{2}}) \geq -\alpha \Delta_{e_j,1} f(o^{j-\frac{1}{2}})$$

It is suffice to prove (4.1). Using the above inequality, when $i_j = 1$,

$$
\begin{aligned}
f(\mathrm{o}^{j-1}) - f(\mathrm{o}^j) &= f(\mathrm{o}^{j-1}) - f(\mathrm{o}^{j-\frac{1}{2}}) - [f(\mathrm{o}^j) - f(\mathrm{o}^{j-\frac{1}{2}})] \\
&= \Delta_{o_j, \mathrm{o}^{j-1}(o_j)} f(\mathrm{o}^{j-\frac{1}{2}}) - \Delta_{e_j,i_j} f(\mathrm{o}^{j-\frac{1}{2}}) \\
&\leq \Delta_{o_j, \mathrm{o}^{j-1}(o_j)} f(\mathrm{s}^{j-1}) - \Delta_{e_j,i_j} f(\mathrm{o}^{j-\frac{1}{2}}) \\
&\leq \Delta_{e_j,i_j} f(\mathrm{s}^{j-1}) - \Delta_{e_j,i_j} f(\mathrm{o}^{j-\frac{1}{2}}) \\
&\leq \Delta_{e_j,i_j} f(\mathrm{s}^{j-1}) + \frac{1}{\alpha}\Delta_{e_j,2} f(\mathrm{o}^{j-\frac{1}{2}}) \\
&\leq \Delta_{e_j,i_j} f(\mathrm{s}^{j-1}) + \frac{1}{\alpha}\Delta_{e_j,2} f(\mathrm{s}^{j-1}) \\
&\leq (1 + \frac{1}{\alpha})\Delta_{e_j,i_j} f(\mathrm{s}^{j-1})
\end{aligned}
$$

When $i_j = 2$,

$$
\begin{aligned}
f(\mathrm{o}^{j-1}) - f(\mathrm{o}^j) &\leq \Delta_{e_j,i_j} f(\mathrm{s}^{j-1}) - \Delta_{e_j,1} f(\mathrm{o}^{j-\frac{1}{2}}) \\
&\leq \Delta_{e_j,i_j} f(\mathrm{s}^{j-1}) + \alpha\Delta_{e_j,1} f(\mathrm{o}^{j-\frac{1}{2}}) \\
&\leq \Delta_{e_j,i_j} f(\mathrm{s}^{j-1}) + \alpha\Delta_{e_j,1} f(\mathrm{s}^{j-1}) \\
&\leq (1+\alpha)\Delta_{e_j,i_j} f(\mathrm{s}^{j-1})
\end{aligned}
$$

Since $\alpha \in (0, 1]$, $1 + \alpha \leq 1 + \frac{1}{\alpha}$, we can combine two cases and deduce

$$
f(\mathrm{o}^{j-1}) - f(\mathrm{o}^j) \leq (1 + \frac{1}{\alpha})[f(\mathrm{s}^j) - f(\mathrm{s}^{j-1})]
$$

Hence,

$$
f(\mathrm{o}) - f(\mathrm{s}) = \sum_{j=1}^{r}(f(\mathrm{o}^{j-1}) - f(\mathrm{o}^j)) \leq \sum_{j=1}^{r}(1 + \frac{1}{\alpha})(f(\mathrm{s}^j) - f(\mathrm{s}^{j-1})) = (1 + \frac{1}{\alpha})f(\mathrm{s})
$$

which means $f(\mathrm{s}) \geq \frac{\alpha}{2\alpha+1} f(\mathrm{o})$. When $\alpha = 1$, then f is bisubmodular function and the approximation ratio is $\frac{1}{3}$ which matches the result of [(12)]. It is easy to see that the running time of the non-monotone case is the same as the monotone case. □

5 Conclusion

In this manuscript, we study the maximization of α-bisubmodular function subject to matroid constraint, designing $\frac{1}{2}$ and $\frac{\alpha}{2\alpha+1}$ for monotone and non-monotone cases, respectively. We also give that the running time of proposed algorithm is $O(rn(IO + EO))$ for both cases.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Badanidiyuru A, Vondrák J. (2014). Fast algorithms for maximizing submodular functions[C]//Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics: 1497-1514.

[2] Calinescu G, Chekuri C, Pal M, et al. (2011). Maximizing a monotone submodular function subject to a matroid constraint[J]. SIAM Journal on Computing, 40(6): 1740-1766.

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

[4] Fisher M L, Nemhauser G L, Wolsey L A. (1978). An analysis of approximations for maximizing submodular set functions—II[M]//Polyhedral combinatorics. Springer, Berlin, Heidelberg: 73-87.

[5] Fujishige S, Tanigawa S, Yoshida Y. (2014), Generalized skew bisubmodularity: A characterization and a min–max theorem[J]. Discrete Optimization, 12: 1-9.

[6] Fujishige S, Tanigawa S. (2018). Polynomial combinatorial algorithms for skew-bisubmodular function minimization[J]. Mathematical Programming, 171: 87-114.

[7] Huber A, Krokhin A, Powell R. (2014). Skew bisubmodularity and valued CSPs[J]. SIAM Journal on Computing, 43(3): 1064-1084.

[8] Iwata S, Tanigawa S, Yoshida Y. (2013). Bisubmodular Function Maximization and Extensions[J]. Technical report, METR, University of Tokyo.

[9] Iwata S, Tanigawa S, Yoshida Y. (2016). Improved approximation algorithms for k-submodular function maximization[C]//Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics: 404-413.

[10] Nemhauser G L, Wolsey L A, Fisher M L. (1978). An analysis of approximations for maximizing submodular set functions—I[J]. Mathematical programming, 14(1): 265-294.

[11] Nemhauser G L, Wolsey L A. (1978). Best algorithms for approximating the maximum of a submodular set function[J]. Mathematics of operations research, 3(3): 177-188.

[12] Ohsaka N, Yoshida Y. (2015). Monotone k-submodular function maximization with size constraints[J]. Advances in Neural Information Processing Systems, 28.

[13] Oshima H. (2021). Improved randomized algorithm for k-submodular function maximization[J]. SIAM Journal on Discrete Mathematics, 35(1): 1-22.

[14] Sakaue S. (2017). On maximizing a monotone k-submodular function subject to a matroid constraint[J]. Discrete Optimization, 23: 105-113.

[15] Shi G, Gu S, Wu W. (2021). k-Submodular maximization with two kinds of constraints[J]. Discrete Mathematics, Algorithms and Applications, 13(04): 2150036.

[16] Sviridenko M. (2004). A note on maximizing a submodular set function subject to a knapsack constraint[J]. Operations Research Letters, 32(1): 41-43.

[17] Tang Z, Wang C, Chan H. (2022). On maximizing a monotone k-submodular function under a knapsack constraint[J]. Operations Research Letters, , 50(1): 28-31.

[18] Ward J, Živný S. (2016). Maximizing k-submodular functions and beyond[J]. ACM Transactions on Algorithms (TALG), 12(4): 1-26.

-