TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 04/22 pp46–55
DOI: 10.26599/TST.2022.9010013
Volume 29, Number 1, February 2024

Maximizing Submodular+Supermodular Functions Subject to a Fairness Constraint

Zhenning Zhang, Kaiqiao Meng, Donglei Du, and Yang Zhou*

Abstract: We investigate the problem of maximizing the sum of submodular and supermodular functions under a fairness constraint. This sum function is non-submodular in general. For an offline model, we introduce two approximation algorithms: A greedy algorithm and a threshold greedy algorithm. For a streaming model, we propose a one-pass streaming algorithm. We also analyze the approximation ratios of these algorithms, which all depend on the total curvature of the supermodular function. The total curvature is computable in polynomial time and widely utilized in the literature.

Key words: submodular function; supermodular function; fairness constraint; greedy algorithm; threshold greedy algorithm; streaming algorithm

1 Introduction

Submodular maximization has become one of the most attractive problems in combinatorial optimization, since the greedy algorithm[1] was introduced by Nemhauser et al. The diminishing return of a set function is regarded as an equivalent definition of submodularity, which has been well applied in economics and artificial intelligence. In Ref. [1], Nemhauser et al. showed that the approximation ratio, $1-1/e$ of their greedy algorithm, cannot be improved for maximizing monotone nonnegative submodular function unless $P=NP$. Many new results have been arising. Except the greedy algorithm, many other algorithms, such as local search[2], continuous greedy algorithm[3], adaptive algorithms[4], and streaming algorithms[5], have been proposed. Various constraints, such as cardinality[1, 5], knapsack[6, 7], matroid[3], and fairness[8, 9], have also been considered. A number of models, including non-submodular models[10–12], models on the lattice[12, 13], off-line[14], and online models[15], have been investigated.

Submodular maximization has been extensively studied and widely applied. However, most practical problems are non-submodular in nature[10–12]. Bian et al. introduced the submodularity ratio[11] to describe the difference between a non-submodular function and a submodular function. They proposed a greedy algorithm and attained the first tight approximation ratio with respect to the submodularity ratio. However, the submodularity ratio of a non-submodular function is highly related to the ground set and cannot be computed in polynomial time. Bai and Bilmes[10] investigated a non-submodular function with special structures; that is, the non-submodular function is a submodular function plus a supermodular function. By introducing the polynomial-time computable parameters, named total curvatures[16], they obtained a constant approximation ratio for the greedy algorithm.

Fairness constraint[9] is a new constraint that was proposed by Wang et al., and it is considered as a special case of matroid partition constraints. When we investigate the maximization of the submodular

• Zhenning Zhang and Kaiqiao Meng are with Beijing Institute for Scientific and Engineering Computing, Beijing University of Technology, Beijing 100124, China. E-mail: zhangzhenning@bjut.edu.cn; mengkaiqiao@126.com.
• Donglei Du is with the Faculty of Management, University of New Brunswick, Fredericton E3B 5A3, Canada. E-mail: ddu@unb.ca.
• Yang Zhou is with School of Mathematics and Statistics, Shandong Normal University, Jinan 250014, China. E-mail: zhouyang@sdnu.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2022-02-18; revised: 2022-03-14; accepted: 2022-04-20

function with a cardinality constraint, the solution would probably be obtained in a subset with the same properties. Taking the Olympic Games as an example, if we put male and female athletes in the same competitions, then male athletes have a high probability winning all the medals. Thus, fairness constraint is significant for practical problems[17]. In Ref. [9], the fairness constraint is given in the form $|S \cap E_j| \leqslant k_j$ with $\sum_{j=1}^{l} k_j = k$, where S is a feasible solution, E_j $(j = 1, \ldots, l)$ is the partition of the ground set E with $E_i \cap E_j = \varnothing$ $(i \neq j)$ and $\cup_{j=1}^{l} E_j = E$, k_j is a known budget given as $k_j = \dfrac{|E_j|}{|E|}k$ and k is the total budget. The authors mainly considered a streaming model and gave multi-pass and one-pass streaming algorithms.

In Ref. [8], Halabi et al. proposed another form of fairness constraint as $q_j \leqslant |S \cap E_j| \leqslant p_j$, and $\sum_{j=1}^{l} q_j \leqslant k$, where p_j and q_j are constants. Their streaming algorithm was designed by invoking an approximation algorithm for the submodular maximization with a matroid constraint.

Here, we mainly focus on the maximization of a submodular function plus a supermodular function with a fairness constraint. We design two offline algorithms and one streaming algorithm. First, using the greedy algorithm alone, we choose the element with the largest marginal increment at each step. By utilizing the total curvature of the supermodular function, we analyze the approximation ratio of the greedy algorithm. If the objective function reduces to a submodular function, we obtain the same approximation ratio given in Ref. [9]. Second, we design the threshold greedy algorithm to overcome this problem. We pick the element if its marginal increment is larger than a given threshold. We use the thresholds to reduce the inquiry complexity. Finally, a one-pass streaming algorithm is provided. However, we cannot obtain the total curvature of the supermodular function in advance. Here we guess it by dividing the interval $[0, 1)$ equally and returning the solution with the largest objective value at the end. We also give the guarantee, memory complexity, and inquiry complexity of the algorithm.

The rest of the paper is organized as follows. In Section 2, we describe the investigated problem, and introduce some notations and conclusions used throughout the paper. In Section 3, we provide a greedy algorithm and analyze its approximation ratio. In Section 4, to improve the inquiry complexity, we introduce the threshold greedy algorithm and give its guarantee. In Section 5, a one-pass streaming algorithm with post-processing is proposed, and its properties are analyzed.

2 Preliminary

In this section, we introduce some notations, definitions, and properties. We also give a description of the problem we focus on.

Given a ground set $E = \{e_1, \ldots, e_n\}$, the monotone non-decreasing function is defined as $f(S) \leqslant f(T)$ for any $S \subseteq T$, S and $T \in 2^E$. Let $[n]$ denote the set $[n] = \{1, 2, \ldots, n\}$.

The definitions of submodularity and its equivalent form are given as follows:

Definition 1[13] A set function $f(\cdot) : 2^E \to \mathbf{R}_+$ is submodular if for any subsets $S, T \in 2^E$, the equation

$$f(S) + f(T) \geqslant f(S \cup T) + f(S \cap T)$$

holds.

Meanwhile, the most useful equivalent form is the Diminishing Return submodularity (DR-submodularity) which is defined as follows:

Definition 2[13] If a set function $f\colon 2^E \to \mathbf{R}_+$ satisfies

$$f(S \cup \{e\}) - f(S) \geqslant f(T \cup \{e\}) - f(T)$$

for every S and $T \in 2^E$, $S \subseteq T$, $e \in E \setminus T$, the function $f(\cdot)$ is DR-submodular.

Similarly, the supermodularity of a set function is defined.

Definition 3 Given a set function $g\colon 2^E \to \mathbf{R}_+$, we say that g is supermodular if for any S and $T \in 2^E$, $S \subseteq T$, $e \in E \setminus T$,

$$g(S \cup \{e\}) - g(S) \leqslant g(T \cup \{e\}) - g(T),$$

which is equivalent to

$$g(S) + g(T) \leqslant g(S \cup T) + g(S \cap T)$$

for any subsets S and $T \in 2^E$.

We investigate the maximization of a monotone submodular function plus a supermodular function[10] with a fairness constraint[9]. A given ground set E is divided into l subsets $E_1, \ldots, E_l$, satisfying $E_i \cap E_j = \varnothing$ and $\cup_{j=1}^{l} E_j = E$. The objective function $h(X) = f(X) + g(X)$ is a monotone nonnegative non-submodular function with $h(\varnothing) = 0$, where $f(X)$ and $g(X)$ are a monotone nonnegative submodular function and a monotone nonnegative supermodular function on E, respectively. The target of the problem is to get a subset $X \in 2^E$ to maximize the objective function $h(X)$ with the constraints $|X \cap E_i| \leqslant k_i$ for all $i \in [l]$ and $\sum_{i=1}^{l} k_i = k$, where k_i, $i \in [l]$ are known positive integers. Thus, the feasible solution X with $|X| \leqslant k$

chooses at most k_i elements in each subset E_i ($i \in [l]$). The formula of the problem is as follows:

$$\max_{X \in 2^E, |X \cap E_i| \leqslant k_i, i \in [l],} h(X) \tag{1}$$

For two subsets S and T, we denote $h(S|T) := h(S \cup T) - h(T)$ as the marginal value of a set S when it is added to another set T.

To measure the non-submodular function $h(X)$ with polynomial-time computable parameters, we introduce the total curvatures of submodular and supermodular functions.

Definition 4[10, 16] Given a monotone nonnegative submodular function $f\colon 2^E \to \mathbf{R}_+$, the total curvature of $f(\cdot)$ is defined as follows:

$$c_f = 1 - \min_{e \in E} \frac{f(E) - f(E \setminus \{e\})}{f(\{e\})},$$

where we suppose that $f(\{e\}) \neq 0$ for any $e \in E$.

From the definition it is obvious to see that $c_f \in [0, 1)$. Moreover, the function is modular if and only if $c_f = 0$.

Proposition 1[10] Let $g(X)$ be supermodular. Its dual function $g(E) - g(E \setminus \{X\})$ is submodular with respect to X.

Proof For any $X \subseteq Y, e \in E \setminus Y$, from the definition of supermodularity, we obtain

$$g(X \cup \{e\}) - g(X) \leqslant g(Y \cup \{e\}) - g(Y).$$

Furthermore, let $l(X) = g(E) - g(E \setminus X)$, we can prove the following:

$$\begin{aligned}
&l(X \cup \{e\}) - l(X) = \\
&g(E) - g(E \setminus (X \cup \{e\})) - (g(E) - g(E \setminus X)) = \\
&g(E \setminus X) - g(E \setminus (X \cup \{e\})) \geqslant \\
&g(E \setminus Y) - g(E \setminus (Y \cup \{e\})) = \\
&g(E) - g(E \setminus (Y \cup \{e\})) - (g(E) - g(E \setminus Y)) = \\
&l(Y \cup \{e\}) - l(Y),
\end{aligned}$$

where the first inequality holds for the supermodularity of $g(\cdot)$.

Therefore, $l(X) = g(E) - g(E \setminus X)$ is a DR-submodular function with respect to X. ■

Thus the total curvature of a supermodular function can be derived from its dual submodular function as follows.

Definition 5[10] Let $g(\cdot)$ be a monotone nonnegative supermodular function on E, the total curvature of $g(\cdot)$ is defined in the following:

$$c^g = c_{g(E) - g(E \setminus X)} = 1 - \min_{e \in E} \frac{g(\{e\})}{g(E) - g(E \setminus \{e\})}.$$

Similar to the case of submodularity, we can obtain that $c^g \in [0, 1)$ and $g(\cdot)$ is modular if and only if $c^g = 0$.

The advantage of these parameters is that they can be obtained by at most $2n + 1$ oracle queries [10, 14].

By utilizing the total curvatures, we have the following properties of the objective function $h(\cdot)$.

Lemma 1[10] Let $f(\cdot)$ be a monotone nonnegative submodular function with total curvature c_f, and $g(\cdot)$ be a monotone nonnegative supermodular function with total curvature c^g. Then, for the function $h(\cdot) = f(\cdot) + g(\cdot)$ and any two subsets S and $T \subseteq E$, satisfying $S \subseteq T$ and $e \in E \setminus T$,

$$\left(1 - c_f\right)\left(h(S \cup \{e\}) - h(S)\right) \leqslant h(T \cup \{e\}) - h(T) \tag{2}$$

$$\left(1 - c^g\right)\left(h(T \cup \{e\}) - h(T)\right) \leqslant h(S \cup \{e\}) - h(S) \tag{3}$$

Proof From Definition 2, we have

$$\left(1 - c_f\right) f(\{e\}) \leqslant f(E) - f(E \setminus \{e\}) \tag{4}$$

From submodularity, we obtain the expression below:

$$f(E) - f(E \setminus \{e\}) \leqslant f(T \cup \{e\}) - f(T) \tag{5}$$

$$f(S \cup \{e\}) - f(S) \leqslant f(\{e\}) \tag{6}$$

where S and $T \subseteq E$. If $S \subseteq T$, combining Formulas (4)–(6), we achieve

$$\begin{aligned}
&\left(1 - c_f\right)\left(f(S \cup \{e\}) - f(S)\right) \leqslant \\
&\left(1 - c_f\right) f(\{e\}) \leqslant \\
&f(E) - f(E \setminus \{e\}) \leqslant \\
&f(T \cup \{e\}) - f(T)
\end{aligned} \tag{7}$$

Correspondingly, for function $g(\cdot)$ there holds

$$\begin{aligned}
&\left(1 - c_f\right)\left(g(S \cup \{e\}) - g(S)\right) \leqslant \\
&g(S \cup \{e\}) - g(S) \leqslant \\
&g(T \cup \{e\}) - g(T)
\end{aligned} \tag{8}$$

where $S \subseteq T$. Then Formula (2) can be derived by combining Formulas (7) and (8). By Definition 2, we can obtain

$$\left(1 - c^g\right)\left(g(E) - g(E \setminus \{e\})\right) \leqslant g(\{e\}) \tag{9}$$

Moreover, by the supermodularity of function $g(\cdot)$, we have

$$g(T \cup \{e\}) - g(T) \leqslant g(E) - g(E \setminus \{e\}) \tag{10}$$

$$g(\{e\}) \leqslant g(S \cup \{e\}) - g(S) \tag{11}$$

where S and $T \subseteq E$. If $S \subseteq T$, then together with Formulas (9)–(11), we obtain

$$\begin{aligned}
&\left(1 - c^g\right)\left(g(T \cup \{e\}) - g(T)\right) \leqslant \\
&\left(1 - c^g\right)\left(g(E) - g(E \setminus \{e\})\right) \leqslant \\
&g(\{e\}) \leqslant g(S \cup \{e\}) - g(S)
\end{aligned} \tag{12}$$

For the submodular function $f(\cdot)$ and there holds

$$(1-c^g)\,(f(T\cup\{e\})-f(T))\leqslant$$
$$f(T\cup\{e\})-f(T)\leqslant$$
$$f(S\cup\{e\})-f(S) \tag{13}$$

where $S\subseteq T$. Together with Formula (12), we finally finish the proof. ■

3 Greedy Algorithm

First, we use the greedy algorithm to investigate Formula (1). In every step, the algorithm enumerates all the elements in the remaining ground set to find the maximum one with the largest marginal increment. Then, the subset to which the selected element belongs is checked. If the constraint of the subset has not reached the capacity, the element is added to the solution set, and is removed away from the ground set. If the constraint of the subset has already reached the capacity, the element cannot be added to the solution set and we delete all the elements in this subset from the ground set. This procedure is repeated until the remaining ground set is empty. Meanwhile, all the constraints are tight. Algorithm 1 gives the pseudo code of this idea.

To analyze the approximation ratio of Algorithm 1, we introduce some notations. Let O denote an optimal solution set of Formula (1) and $O_j = O\cap E_j$ be the optimal solution in the subset E_j ($j\in[l]$).

Theorem 1 The performance guarantee of Algorithm 1 is $(1-c^g)/(2-c^g)$. The inquiry complexity of Algorithm 1 is $O(nk)$.

Proof The final solution X obtained by Algorithm 1 is divided into $X_1, X_2, \ldots, X_l$ according to the subsets $E_1, E_2, \ldots, E_l$. Given that $|O_j| = |X_j| = k_j$, we construct bijections $\pi_j : O_j \to X_j$ from O_j to X_j, $j = 1, 2, \ldots, l$. For $o \in O_j \cap X_j$, the bijection maps o to itself. The subset $X_j \setminus O_j = \{x_{j_1}, x_{j_2}, \ldots, x_{j_{q_j}}\}$ ($q_j \leqslant k_j$) is labeled in the order of the elements added by Algorithm 1. The set $O_j \setminus X_j$ is ordered $\{o_{j_1}, o_{j_2}, \ldots, o_{j_{q_j}}\}$ arbitrary. For $o_{j_i} \in O_j \setminus X_j$, the bijections are $\pi_j(o_{j_i}) = x_{j_i}$, $i = 1, 2, \ldots, q_j$. Let X^{j_i} denote the initial solution of the iteration in which the element x_{j_i} is added. According to Algorithm 1, the equation $h(o_{j_i}|X^{j_i}) \leqslant h(x_{j_i}|X^{j_i})$ holds. Thus, we have

$$h(O)-h(X)\leqslant h(O\cup X)-h(X)\leqslant$$
$$\frac{1}{1-c^g}\sum_{j=1}^{l}\sum_{i=1}^{q_j}h(o_{j_i}|X^{j_i})\leqslant$$
$$\frac{1}{1-c^g}\sum_{j=1}^{l}\sum_{i=1}^{q_j}h(x_{j_i}|X^{j_i})\leqslant$$
$$\frac{1}{1-c^g}h(X).$$

By rearrangement, the above inequality can be written as follows:

$$h(X)\geqslant\frac{1-c^g}{2-c^g}h(O) \tag{14}$$

The algorithm needs to query the marginal increment of at most n elements in each round, and enumerates k rounds. Thus, we obtain the conclusions in Theorem 1. ■

Remark 1 If $c^g = 0$, then $g(\cdot)$ is a modular function. At this point, Algorithm 1 is guaranteed by $1/2$.

Algorithm 1 **Greedy**$(\varnothing, E)$

input: function $h(\cdot)$, ground set E, partitions $E_1, E_2, \ldots, E_l \subseteq E$, total budget $k \in \boldsymbol{N}$, budgets of each partitions $k_1, k_2, \ldots, k_l \in \boldsymbol{N}$
output: X

```
X ← ∅, S ← E;
while S ≠ ∅ do
    e ← arg max_{e∈S} h(X ∪ {e}) − h(X);
    Find e ∈ E_j;
    if |X ∩ E_j| < k_j then
        X ← X ∪ {e}, S ← S \ {e};
    else
        S ← S \ E_j;
    end if
end while
return X
```

4 Threshold Greedy Algorithm

In this section, to reduce the inquiry complexity, we propose the threshold greedy algorithm for Formula (1). The main idea is as follows: when the marginal increment of an element for the current solution is larger than some threshold, it will be added to the solution. Intuitively, the threshold is related to the OPTimal value (OPT). Moreover, the threshold can be considered as one over k of OPT. The OPT can be estimated by using Formula (3),

$$d\leqslant h(O)\leqslant\frac{1}{1-c^g}\sum_{o\in O}h(o)\leqslant\frac{kd}{1-c^g} \tag{15}$$

where $d = \max_{e\in E} h(\{e\})$ is the maximum value of any singleton element. Thus, the one over k of OPT falls in the interval $\left[\epsilon\frac{d}{k}, \frac{d}{1-c^g}\right]$, where ϵ is a small positive constant. We reduce from $\frac{d}{1-c^g}$ by the order of times the geometric progression $(1-\epsilon)^{i-1}$,

$\left(i = 1, \ldots, O\left(\frac{\log k}{\epsilon}\right)\right)$ and consider all these values as thresholds. For each threshold, we enumerate all elements in the ground set. If the marginal increment of an element is larger than the threshold and the corresponding constraint has not reached its upper bound, then the element is added to the solution. More details are shown in Algorithm 2.

To analyze the approximation ratio of the algorithm, we consider the upper bound of the marginal increment for an element which is in the optimal solution but not in the current solution. Suppose that Algorithm 2 is terminated as the threshold lower than $\epsilon\frac{d}{k}$. Thus, the solution is updated at most $O\left(\frac{n\log k}{\epsilon}\right)$ times. Let $p = O\left(\frac{n\log k}{\epsilon}\right)$ denote the final step of the iteration. We then have the following lemma.

Lemma 2 Let x_i ($x_i \in E_j (j \in [l])$) be processed at the i-th iteration ($i = 1, \ldots, p$) in Algorithm 2 and X^{i-1} with $|X^{i-1} \cap E_j| < k_j$ be the initial solution for the i-th iteration. $X^i = X^{i-1} \cup \{x_i\}$ ($i = 1, \ldots, p$) is the solution after the i-th iteration. For any $o \in O_j \setminus X^{i-1}$ ($j = 1, \ldots, l$), the relationship between the increment of o to X^{i-1} and the gain at the i-th iteration is given as follows:

$$h(o|X^{i-1}) \leqslant \frac{1}{(1-\varepsilon)(1-c^g)} h(x_i|X^{i-1}) \tag{16}$$

where o and x_i are in the same partition $E_j (j = 1, \ldots, l)$.

Proof First, suppose that the threshold is $\theta = \frac{d}{1-c^g}$. Thus, the increment is

$$h(x_i|X^{i-1}) \geqslant \theta = \frac{d}{1-c^g} > \frac{1-\varepsilon}{1-c^g} d \geqslant \frac{1-\varepsilon}{1-c^g} h(o) \geqslant (1-\varepsilon) h(o|X^{i-1}) \geqslant (1-\varepsilon)(1-c^g) h(o|X^{i-1}),$$

where $o \in O \setminus X^{i-1}$.

Second, for $\theta < \frac{d}{1-c^g}$, suppose that the last threshold is $\frac{\theta}{1-\varepsilon}$, and X^m ($X^m \subseteq X^{i-1}$) is the last feasible solution with respect to the threshold $\frac{\theta}{1-\varepsilon}$. Then, all the elements in the ground set E have been read at least once. For any $o \in O_j \setminus X^{i-1}$, let X^{o-1} ($|X^{o-1} \cap E_j| < k_j$) be the initial solution for the iteration to consider the element o. As o has not been added, we therefore have

$$h(o|X^{o-1}) \leqslant \frac{\theta}{1-\varepsilon} \tag{17}$$

From Formula (3), we have

$$h(o|X^{i-1}) \leqslant \frac{1}{1-c^g} h(o|X^{o-1}) \tag{18}$$

where $X^{o-1} \subseteq X^m \subseteq X^{i-1}$. Meanwhile, from Algorithm 2, at the i-th iteration, we have

$$h(x_i|X^{i-1}) \geqslant \theta \tag{19}$$

Combining Formulas (17)–(19) immediately yields Lemma 4. ■

Theorem 2 Algorithm 2 is guaranteed by $\frac{(1+c^g)^2(1-\epsilon)^2}{(1+c^g)^2(1-\epsilon)^2+1}$. The inquiry complexity of Algorithm 2 is $O\left(\frac{n\log k}{\epsilon}\right)$.

Proof From the algorithm, the final output solution X^p satisfies $|X^p| = k$ or $|X^p| < k$. Let $X_j^p = X^p \cap E_j$, $j \in [l]$ be the partition of X^p based on E_j ($j \in [l]$).

(1) For the case $|X^p| = k$, each constraint satisfies $|X_j^p| = k_j$. We first label the elements in the set $X_j^p = \{x_{j_1}, \ldots, x_{j_{k_j}}\}$ according to the order of the element

Algorithm 2 Threshold greedy

input: submodular function $f(\cdot)$, supermodular function $g(\cdot)$, $h(\cdot) = f(\cdot) + g(\cdot)$, ground set E, partitions $E_1, E_2, \ldots, E_l \subseteq E$, total budget $k \in \mathbf{N}$, budgets for each partitions $k_1, k_2, \ldots, k_l \in \mathbf{N}$, parameter $\epsilon \in (0, 1)$
output: X

```
c^g = 1 − min_{e∈E} g({e}) / (g(E) − g(E\{e}));
X ← ∅, S ← E, d ← max_{e∈E} h({e});
for (θ = d/(1−c^g); θ ⩾ ε d/k; θ ← (1−ε)θ) do
    for all e ∈ E do
        Find e ∈ E_j;
        if |X ∩ E_j| < k_j then
            if h(X ∪ {e}) − h(X) ⩾ θ then
                X ← X ∪ {e}, S ← S \ {e};
            else
                X ← X, S ← S;
            end if
        else
            S ← S \ E_j;
        end if
    end for
    if |X| = k then
        break;
    end if
end for
return X
```

added. As $|O_j| = |X_j^p| = k_j$, we construct bijections π_j ($i \in [l]$) from O_j to X_j^p similar as the bijections in the proof of Theorem 1. As a result, we obtain the following:

$$\pi_j(o_{j_i}) = \begin{cases} o_{j_i}, & \text{if } o_{j_i} \in O_j \cap X_j^p; \\ x_{j_i}, & \text{otherwise} \end{cases} \tag{20}$$

Let X^{j_i-1} be the initial solution of the iteration in which x_{j_i} is added. Thus, we have

$$\begin{aligned} &h(O) - h(X^p) \leqslant \\ &h(O \cup X^p) - h(X^p) \leqslant \\ &\frac{1}{1-c^g} \sum_{j=1}^{l} \sum_{o_{j_i} \in O_j \setminus X^p} h(o_{j_i}|X^{j_i-1}) \leqslant \\ &\frac{1}{(1-c^g)^2(1-\epsilon)} \sum_{j=1}^{l} \sum_{x_{j_i} \in X_j^p \setminus O_j} h(x_{j_i}|X^{j_i-1}) \leqslant \\ &\frac{1}{(1-c^g)^2(1-\epsilon)} h(X^p) \end{aligned} \tag{21}$$

The first inequality holds by monotonicity, and the second and the third inequalities hold via Formula (3) and Lemma 4, respectively. By rearrangement, we obtain

$$h(X^p) \geqslant \frac{(1+c^g)^2(1-\epsilon)}{(1+c^g)^2(1-\epsilon)+1} h(O) \tag{22}$$

(2) In the case $|X^p| < k$, some constraints satisfy $|X_j^p| < k_j$, and others satisfy $|X_j^p| = k_j$. We label the index sets $I_1 = \{j \,|\, |X_j^p| < k_j, j \in [l]\}$ and $I_2 = \{j \,|\, |X_j^p| = k_j, j \in [l]\}$.

According to Algorithm 2, we have

$$h(o|X^p) < \frac{\epsilon d}{k} \tag{23}$$

for any $o \in O_j \setminus X^p$ with $j \in I_1$. Thus, we get

$$\begin{aligned} &\sum_{j \in I_1} \sum_{o \in O_j \setminus X^p} h(o|X^p) \leqslant \epsilon h(X^p) \leqslant \\ &\frac{\epsilon}{(1-c^g)(1-\epsilon)} h(X^p) \end{aligned} \tag{24}$$

For the case of $o \in O_j \setminus X^p$ with $j \in I_2$, we construct bijections similar to Eq. (20). Thus, we have

$$\begin{aligned} &\sum_{j \in I_2} \sum_{o_{j_i} \in O_j \setminus X^p} h(o_{j_i}|X^{j_i-1}) \leqslant \\ &\frac{1}{(1-c^g)(1-\epsilon)} \sum_{j \in I_2} \sum_{x_{j_i} \in X_j^p \setminus O_j} h(x_{j_i}|X^{j_i-1}) \leqslant \\ &\frac{1}{(1-c^g)(1-\epsilon)} h(X^p) \end{aligned} \tag{25}$$

Combining Formulas (24) and (25), we attain in the following:

$$\begin{aligned} &h(O) - h(X^p) \leqslant \\ &h(O \cup X^p) - h(X^p) \leqslant \\ &\frac{1}{1-c^g} \sum_{j \in I_1} \sum_{o \in O_j \setminus X^p} h(o|X^p) + \\ &\frac{1}{1-c^g} \sum_{j \in I_2} \sum_{o_{j_i} \in O_j \setminus X^p} h(o_{j_i}|X^{j_i-1}) \leqslant \\ &\frac{1+\epsilon}{(1-c^g)^2(1-\epsilon)} h(X^p). \end{aligned}$$

By rearrangement, we obtain

$$\begin{aligned} h(X^p) \;\geqslant\; &\frac{(1+c^g)^2(1-\epsilon)}{(1+c^g)^2(1-\epsilon)+1+\epsilon} h(O) \geqslant \\ &\frac{(1+c^g)^2(1-\epsilon)^2}{(1+c^g)^2(1-\epsilon)^2+1} h(O) \end{aligned} \tag{26}$$

Combining Formulas (22) and (26), we achieve the conclusion in Theorem 2. ■

Remark 2 When $c^g = 0$, $h(\cdot)$ is generated to a submodular function. Thus, Algorithm 2 becomes a $(1/2-\epsilon)$-approximation algorithm .

5 Streaming Algorithm

Finally, we attempt to investigate Formula (1) with a big data ground set. We cannot store all data information and calculate them simultaneously. We need a streaming algorithm. Streaming algorithms consider the data one by one, and before the next element reveals, it must make a decision for the current one. Four indexes are used to measure the performance of a streaming algorithm. The approximation ratio and time complexity are formal indexes. The other two are memory complexity and the number of passes of reading the total data. We aspire to attain a one-pass streaming algorithm with a reasonable approximation ratio, time complexity and memory complexity.

Similar to the threshold greedy algorithm, if the marginal increment of the current element is larger than the threshold, then it is added to the solution. We subdivide the estimation interval of the OPT by geometric series, and consider all these values as the threshold. By Formula (15), the bound of the OPT is as follows:

$$\eta \leqslant h(O) \leqslant \frac{kd}{1-c^g} \tag{27}$$

where $\eta = \max[d, h(X)]$.

We subdivide the interval $\left[\eta, \frac{2kd}{1-c^g}\right]$ and obtain a set in the following:

$$\mathcal{H}=\left\{(1+\epsilon)^m|m\in\mathbf{Z},\max\{d,\eta\}\leqslant(1+\epsilon)^m\leqslant\frac{2kd}{1-c^g}\right\}.$$

Suppose that $(1+\epsilon)^m = h(O)$. Then, we attain $m^*=\left\lfloor\dfrac{\log h(O)}{\log(1+\epsilon)}\right\rfloor$. Let $\tau^*=(1+\epsilon)^{m^*}$. Evidently, we have $\tau^*\leqslant h(O)$. Meanwhile, $\tau^*\geqslant\dfrac{h(O)}{1+\epsilon}\geqslant(1-\epsilon)h(O)$. Therefore, the parameter τ^* is contained in $\mathcal{H}$ with $(1-\epsilon)h(O)\leqslant\tau^*\leqslant h(O)$. τ^* can be considered as an estimation of the OPT.

As the element is revealed one by one in streaming algorithms, we cannot obtain the maximum value d of a singleton element in advance. Meanwhile, the feasible solution is updated with the new arriving element. Thus, the estimation interval and the parameter set $\mathcal{H}$ are updated. Finally, the set desired is exactly $\mathcal{H}_n$ with $(1-\epsilon)h(O)\leqslant\tau^*\leqslant h(O)$.

Another difficulty is that c^g is the total curvature of the supermodular function $g(\cdot)$, whose computation needs the entire information of the data set. Thus, we cannot receive it in advance. Given that $c^g\in[0,1)$, we refine the interval $[0,1)$ into $\lfloor 1/\epsilon\rfloor$ copies to simulate the value c^g. The more copies of the interval $[0,1)$ are subdivided, the higher accuracy of the solution can be achieved. Finally, the algorithm returns a solution with the largest value among all these $c^g\in\{0,\epsilon,2\epsilon,\ldots,\lfloor 1/\epsilon\rfloor\epsilon\}$ and τ in $\mathcal{H}_n$. The pseudo code of the one-pass streaming algorithm is presented in Algorithm 3.

In the following lemma, we consider the properties of the solution X_{τ^*} corresponding to τ^* in the set $\mathcal{X}$, where $\mathcal{X}=\{X_\tau,\tau\in\mathcal{H}_n\}$ is the set of all final solutions with respect to the parameters $\tau\in\mathcal{H}_n$ returned by the streaming processing and $(1-\epsilon)h(O)\leqslant\tau^*\leqslant h(O)$. The constraints of X_{τ^*} may result in (1) $|X_{\tau^*}\cap E_j|=k_j$ for all $j\in[l]$, or (2) $|X_{\tau^*}\cap E_j|<k_j$ for all $j\in[l]$, or (3) $|X_{\tau^*}\cap E_j|=k_j$ for some j and $|X_{\tau^*}\cap E_j|<k_j$ for others. One of these three cases may occur.

Lemma 3 For the parameter τ^* with $(1-\epsilon)h(O)\leqslant\tau^*\leqslant h(O)$, if the cardinality of X_{τ^*} is $|X_{\tau^*}|=k$, or $|X_{\tau^*}\cap E_j|<k_j$ for all $j\in[l]$, we have $h(X_{\tau^*})\geqslant\left(1-\dfrac{1}{2(1-c^g)}-\epsilon\right)h(O)$.

Proof By mathematical induction, we get

$$h(X_{\tau^*})\geqslant\frac{\tau^*}{2k}|X_{\tau^*}|.$$

For the case of $|X_{\tau^*}|=k$, it is obvious that

$$h(X_{\tau^*})\geqslant\frac{\tau^*}{2}\geqslant\frac{1}{2}(1-\epsilon)h(O).$$

For the case that $|X_{\tau^*}\cap E_j|<k_j$ for all $j\in[l]$, we have

$$h(O)-h(X_{\tau^*})\leqslant h(O\cup X_{\tau^*})-h(X_{\tau^*})\leqslant\frac{1}{1-c^g}\sum_{o\in O\setminus X_{\tau^*}}h(o|X_{\tau^*})\leqslant\frac{1}{1-c^g}\cdot\frac{\tau^*}{2k}\cdot k\leqslant\frac{1}{2(1-c^g)}h(O),$$

where the third inequality holds by Algorithm 3.

By rearrangement, we get

$$h(X_{\tau^*})\geqslant\left(1-\frac{1}{2(1-c^g)}\right)h(O)\tag{28}$$

From the above, we receive the conclusions in Lemma 5. ■

Algorithm 3 Streaming / submodular plus supermodular

input: $f(\cdot)$, $g(\cdot)$, $h(\cdot)=f(\cdot)+g(\cdot)$, E, $E_1,E_2,\ldots,E_l\subseteq E$, $k\in\mathbf{N}$, and $k_1,k_2,\ldots,k_l\in\mathbf{N}$, ϵ and $\beta\in(0,1)$
output: X

for each $c^g=r\epsilon$, $r=0,\ldots,\lfloor 1/\epsilon\rfloor$ **do**
  $d\leftarrow 0$, $\eta\leftarrow 0$, $M\leftarrow\varnothing$, $Q_j\leftarrow\varnothing$ for $j\in[l]$;
  for each $i\in[n]$ **do**
    $d=\max\{d,h(\{e_i\})\}$;
    Update Q_j with respect to e using reservoir sampling;
    $\mathcal{H}_i=\{(1+\epsilon)^m|m\in\mathbf{Z},\max\{d,\eta\}\leqslant(1+\epsilon)^m\leqslant\frac{2kd}{1-c^g}\}$;
    $X_\tau\leftarrow\varnothing$ for each new appeared τ in $\mathcal{H}_i$;
    Discard X_τ for all $\tau\notin\mathcal{H}_i$;
    Find $e_i\in E_j$;
    for each $\tau\in\mathcal{H}_i$ **do**
      if $|X_\tau\cap E_j|<k_j$ **then**
        if $h(e_i|X_\tau)\geqslant\frac{\tau}{2k}$ **then**
          $X_\tau\leftarrow X_\tau\cup\{e_i\}$;
        else {if $h(e_i|X_\tau)\geqslant\frac{\beta\eta}{2k}$};
          $M\leftarrow M\cup\{e_i\}$;
        end if
      end if
    end for
    $\eta=\max_{\tau\in\mathcal{H}_i}h(X_\tau)$;
  end for
  $\mathcal{X}=\{X_\tau,\tau\in\mathcal{H}_n\}$;
  if $\Omega=\{\tau:X_\tau(E_j)<k_j$ for all $j\in[l],X_\tau\in\mathcal{X}\}\neq\varnothing$ **then**
    $\varphi\leftarrow\min_{\tau\in\Omega}\tau$;
  else
    $\varphi\leftarrow\max_{\tau\in\mathcal{H}_n}\tau$;
  end if
  for each $\tau\leqslant\varphi$ and $\tau\in\mathcal{H}_n$ **do**
    $X\leftarrow X_\tau$, $S\leftarrow M\cup\left(\cup_{j=1}^{l}Q_j\right)$;
    $X_\tau\leftarrow$ Greedy$\left(X_\tau,M\cup\left(\cup_{j=1}^{l}Q_j\right)\right)$;
  end for
  return $X^{c^g}=\arg\max_{\tau\in\mathcal{H}_n}h(X_\tau)$;
end for
return $X=\arg\max_{c^g=0,1,\ldots,\lfloor 1/\epsilon\rfloor\epsilon}h(X^{c^g})$

However, if the constraints of X_{τ^*} are $|X_{\tau^*} \cap E_j| = k_j$ for some j and $|X_{\tau^*} \cap E_j| < k_j$ for others, we cannot achieve the quality of X_{τ^*}. Thus, Algorithm 3 employs a post-processing to improve the solution quality and help us achieve an approximation ratio of the algorithm. We keep some elements during the streaming processing and store them in set M. The marginal increments of these elements are less than the threshold $\frac{\tau}{2k}$ but no less than $\frac{\beta\eta}{2k}$. Set M is considered as the ground set in the post-processing. The solutions X_φ based on the special parameter φ are considered. Two cases are investigated. One is when φ is assigned as the minimum value in set $\Omega = \{\tau : X_\tau(E_j) < k_j \text{ for all } j \in [l], X_\tau \in \mathcal{X}\}$, if $\Omega \neq \varnothing$. If $\Omega = \varnothing$, then φ is the maximum value in set $\mathcal{H}_n$. We consider the properties of the final solution X_φ^f whose initial solution is X_φ in the post-processing.

Theorem 3 Algorithm 3 is guaranteed by $\frac{2(1-c^g)^2-\beta}{2(1-c^g)^2+2+2\epsilon}$, the update time per element in streaming processing is $O\left(\frac{\log k}{\epsilon}\right)$, and the running time for the post-processing is $O\left(\frac{k\log k}{\epsilon}(|M|+k)\right)$.

Proof We first consider the case that $|X_\varphi \cap E_j| < k_j$ for all $j \in [l]$.

In this case, we divide $O \setminus X_\varphi^f$ into two sets: $O_I = \left(O \setminus X_\varphi^f\right) \cap M$, in which the optimal elements are in the set M, but not in the final solution X_φ^f; $O_J = \left(O \setminus X_\varphi^f\right) \cap (E \setminus M)$, in which the optimal elements are deleted directly during streaming processing.

For any $o \in O_I$, we have

$$h(o|X_\varphi^f) \leqslant \frac{1}{1-c^g}h(o|X'_\varphi) \leqslant \frac{1}{1-c^g}h(x|X'_\varphi) \quad (29)$$

where X'_φ ($X'_\varphi \subseteq X_\varphi^f$) is the initial solution when x is added, and x and o are in the same patition. Formula (29) holds both in streaming processing ($h(o|X'_\varphi) < \frac{\varphi}{2k} \leqslant h(x|X'_\varphi)$) and in post processing (the idea of greedy $h(o|X'_\varphi) \leqslant h(x|X'_\varphi)$).

Meanwhile, as the element $o \in O_J$ is deleted during streaming processing, its marginal increment to the corresponding solution X_φ^o is less than $\frac{\beta\eta}{2k}$. Thus, for any $o \in O_J$, we achieve

$$h(o|X_\varphi^f) \leqslant \frac{1}{1-c^g}h(o|X_\varphi^o) < \frac{1}{1-c^g}\cdot\frac{\beta\eta}{2k} \quad (30)$$

Combining Formulas (29) and (30), we have

$$\begin{aligned}
&h(O) - h(X_\varphi^f) \leqslant \\
&h(O \cup X_\varphi^f) - h(X_\varphi^f) \leqslant \\
&\frac{1}{1-c^g}\sum_{o\in O\setminus X_\varphi^f} h(o|X_\varphi^f) = \\
&\frac{1}{1-c^g}\left(\sum_{o\in O_I} h(o|X_\varphi^f) + \sum_{o\in O_J} h(o|X_\varphi^f)\right) \leqslant \\
&\frac{1}{1-c^g}\left(\frac{1}{1-c^g}\sum_{x\in X_\varphi^f} h(x|X'_\varphi) + \frac{1}{1-c^g}\cdot\frac{k\beta\eta}{2k}\right) \leqslant \\
&\frac{1}{(1-c^g)^2}\left(h(X_\varphi^f) + \frac{\beta}{2}h(O)\right) \qquad (31)
\end{aligned}$$

where X'_φ is the initial solution when x is added.

By rearrangement, we obtain

$$h(X_\varphi^f) \geqslant \frac{2(1-c^g)^2-\beta}{2(1-c^g)^2+2}h(O) \quad (32)$$

Second, we investigate the case in which φ is the maximum parameter in $\mathcal{H}_n$. Notably, $\varphi \leqslant \frac{2kd}{1-c^g} \leqslant (1+\epsilon)\varphi$. In such situation, for any solution X_τ ($\tau \in \mathcal{H}_n$), there are at least one element in the set $\{j \mid |X_\tau \cap E_j| = k_j, j \in [l]\}$.

In this case, $O \setminus X_\varphi^f$ is divided into three sets: O_{J_1} in which the optimal elements are deleted in the streaming process since their corresponding partitions have already reached the capacities, where $J_1 = \{j \mid |X_\varphi \cap E_j| = k_j, j \in [l]\}$; $O_{J_2} = \left(O \setminus \left(X_\varphi^f \cup O_{J_1}\right)\right) \cap M$, in which the optimal elements are in the set M but not in the final solution X_φ^f; $O_{J_3} = \left(O \setminus \left(X_\varphi^f \cup O_{J_1}\right)\right) \cap (E \setminus M)$, in which the optimal elements are deleted directly during streaming processing thought the constraints of their corresponding partitions are not tight. However, their marginal increments are less than $\frac{\beta\eta}{2k}$.

For $o \in O_{J_1}$, we have

$$\begin{aligned}
h(o|X_\varphi^f) \leqslant\ & \frac{1}{1-c^g}h(o) \leqslant \frac{1}{1-c^g}d = \\
& \frac{2k}{1-c^g}d\cdot\frac{1}{2k} \leqslant \frac{(1+\epsilon)\varphi}{2k} \leqslant \\
& (1+\epsilon)h(x|X'_\varphi) \leqslant \frac{1+\epsilon}{1-c^g}h(x|X'_\varphi),
\end{aligned}$$

where x is in the same partition with o and X'_φ is the initial solution when x is added. Collect all these x in the set $X_{J_1} = \{x : x \in X_\varphi^f \cap E_j, j \in J_1\}$.

For $o \in O_{J_1}$ and $o \in O_{J_3}$, the relations are similar with Formulas (29) and (30).

Thus, we obtain the calculation as follows:

$$h(O)-h(X_\varphi^f)\leqslant$$
$$h(O\cup X_\varphi^f)-h(X_\varphi^f)\leqslant$$
$$\frac{1}{1-c^g}\sum_{o\in O\setminus X_\varphi^f}h(o|X_\varphi^f)=$$
$$\frac{1}{1-c^g}\left(\sum_{o\in O_{J_1}}+\sum_{o\in O_{J_2}}+\sum_{o\in O_{J_3}}\right)h(o|X_\varphi^f)\leqslant$$
$$\frac{1}{(1-c^g)^2}\Big((1+\epsilon)\sum_{x\in X_{J_1}}h(x|X'_\varphi)+$$
$$\sum_{x\in X_\varphi^f\setminus X_{J_1}}h(x|X'_\varphi)+\frac{k\beta\eta}{2k}\Big)\leqslant$$
$$\frac{1}{(1-c^g)^2}\left((1+\epsilon)h(X_\varphi^f)+\frac{\beta}{2}h(O)\right).$$

By rearrangement, we have

$$h(X_\varphi^f)\geqslant\frac{2(1-c^g)^2-\beta}{2(1-c^g)^2+2+2\epsilon}h(O)\tag{33}$$

Combining Formulas (32) and (33), we obtain the conclusion in Theorem 5.

From $d(1+\epsilon)^m=\dfrac{2kd}{1-c^g}$, the cardinality of the set $\mathcal{H}_n$ is $O\left(\dfrac{\log k}{\epsilon}\right)$. Therefore, in the streaming processing the memory complexity is $O\left(\dfrac{k\log k}{\epsilon}\right)$ for each c^g. At most $(|M|+k)$ elements are stored for the post-processing. As we have to check $\lfloor 1/\epsilon\rfloor$ values of c^g, the total memory of Algorithm 3 is $O\left(\dfrac{k\log k}{\epsilon^2}+\dfrac{|M|}{\epsilon}\right)$. In the streaming process, the inquiry complexity for each element is formed by the number of parameters in set $\mathcal{H}_n$, that is $O\left(\dfrac{\log k}{\epsilon}\right)$. In the post-processing, the algorithm enumerates at most k iterations for $(|M|+k)$ elements with each initial solution. Thus, post-processing needs $O\left(\dfrac{k\log k}{\epsilon}(|M|+k)\right)$ oracle queries for each c^g.

6 Conclusion

This paper investigates the maximization of a non-submodular function which is a submodular function plus a supermodular function. Meanwhile, the constraint about fairness is more practical than the cardinality constraint because it considers more properties of each subset of the ground set. We first give a greedy algorithm which is a high-quality approximation algorithm consumes long running times. To improve its inquiry complexity, we introduce a threshold greedy algorithm. Both of them are offline algorithms. Finally, we give a streaming algorithm with post-processing to overcome the big data problem. We also analyze the quality of the solutions, inquiry complexity and memory complexity of the three algorithms.

Acknowledgment

The first author was supported by the National Natural Science Foundation of China (Nos. 12001025 and 12131003). The second author was supported by the Spark Fund of Beijing University of Technology (No. XH-2021-06-03). The third author was supported by the Natural Sciences and Engineering Research Council of Canada (No. 283106) and the Natural Science Foundation of China (Nos. 11771386 and 11728104). The fourth author is supported by the National Natural Science Foundation of China (No. 12001335).

References

[1] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, An analysis of approximations for maximizing submodular set functions-I, *Math. Program.*, vol. 14, no. 1, pp. 265–294, 1978.

[2] Y. Filmus and J. Ward, Monotone submodular maximization over a matroid via non-oblivious local search, *SIAM J. Comput.*, vol. 43, no. 2, pp. 514–542, 2014.

[3] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, Maximizing a monotone submodular function subject to a matroid constraint, *SIAM J. Comput.*, vol. 40, no. 6, pp. 1740–1766, 2011.

[4] E. Balkanski, A. Rubinstein, and Y. Singer, An exponential speedup in parallel running time for submodular maximization without loss in approximation, in *Proc. 30th Annu. ACM-SIAM Symp. on Discrete Algorithms*, San Diego, CA, USA, 2019, pp. 283–302.

[5] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause, Streaming submodular maximization: Massive data summarization on the fly, in *Proc. 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, New York, NY, USA, 2014, pp. 671–680.

[6] M. Sviridenko, A note on maximizing a submodular set function subject to a knapsack constraint, *Oper. Res. Lett.*, vol. 32, no. 1, pp. 41–43, 2004.

[7] Y. Yoshida, Maximizing a monotone submodular function with a bounded curvature under a knapsack constraint, *SIAM J. Discrete Mathem.*, vol. 33, no. 3, pp. 1452–1471, 2019.

[8] M. El Halabi, S. Mitrovic, A. Norouzi-Fard, J. Tardos, and J. Tarnawski, Fairness in streaming submodular maximization: Algorithms and hardness, in *Proc. 34th Conf. on Neural Information Processing Systems*, Vancouver, Canada, 2020, pp. 13609–13622.

[9] Y. H. Wang, F. Fabbri, and M. Mathioudakis, Streaming submodular maximization with fairness constraints, in *Proc. the Web Conf. 2021*, Ljubljana, Slovenia, 2021, pp. 1340–1350.

[10] W. R. Bai and J. Bilmes, Greed is still good: Maximizing monotone submodular+supermodular (BP) functions, in *Proc. 35th Int. Conf. on Machine Learning*, Stockholmsmässan, Stockholm, Sweden, 2018, pp. 304–313.

[11] A. A. Bian, J. M. Buhmann, A. Krause, and S. Tschiatschek, Guarantees for greedy maximization of non-submodular functions with applications, in *Proc. 34th Int. Conf. on Machine Learning*, Sydney, Australia, 2017, pp. 498–507.

[12] A. Kuhnle, J. D. Smith, V. G. Crawford, and M. T. Thai, Fast maximization of non-submodular, monotonic functions on the integer lattice, in *Proc. 35th Int. Conf. on Machine Learning*, Stockholmsmässan, Sweden, 2018, pp. 2791–2800.

[13] T. Soma and Y. Yoshida, Maximizing monotone submodular functions over the integer lattice, *Math. Program.*, vol. 172, nos. 1&2, pp. 539–563, 2018.

[14] B. Goldengorin, Maximization of submodular functions: Theory and enumeration algorithms, *Eur. J. Oper. Res.*, vol. 198, no. 1, pp. 102–112, 2009.

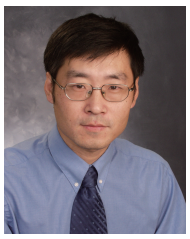
[15] N. Buchbinder, M. Feldman, and R. Schwartz, Online submodular maximization with preemption, *ACM Trans. Algorithms*, vol. 15, no. 3, p. 30, 2019.

[16] M. Conforti and G. Cornuéjols, Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the Rado-Edmonds theorem, *Discrete Appl. Math.*, vol. 7, no. 3, pp. 251–274, 1984.

[17] S. Corbett-Davies, E. Pierson, A. Feller, S. Goel, and A. Huq, Algorithmic decision making and the cost of fairness, in *Proc. 23rd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Halifax, Canada, 2017, pp. 797–806.

Zhenning Zhang received the PhD degree from Beijing Institute of Technology, China in 2010. Currently, she is an associate professor at Beijing University of Technology. Her research interests include combinatorial optimization, approximation algorithm, submodular maximization, and machine learning.

Kaiqiao Meng is currently an undergraduate student at Beijing University of Technology, China. Her research interests include information mathematics and computing science.



Donglei Du received the PhD degrees from Chinese Academy of Sciences, China in 1996, and University of Texas at Dallas, USA in 2003. Currently, he is a professor in operations research at the Faculty of Management (FOM), University of New Brunswick (UNB), Canada. His main research interests are quantitative investment management, combinatorial optimization, approximations algorithms, robust optimization, social network analysis, algorithmic game theory, supply chain management, and facility location, and machine scheduling.

Yang Zhou received the PhD degree in applied mathematics from The Hong Kong Polytechnic University, Hong Kong, China in 2014. He is currently an associate professor at the School of Mathematics and Statistics, Shandong Normal University, Jinan, China. His current research interests include combinatorial optimization, approximation algorithms, large-scale optimization problems, and their applications.