

236621 - Algorithms for Submodular Optimization

Roy Schwartz

April 8, 2019

Abstract

1 Introduction

We are looking on $f : 2^N \rightarrow \mathbb{R}$ for some set $N = \{1, \dots, n\}$

Definition 1.1. f is submodular if

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B) \quad (1)$$

Definition 1.2. Return of u wrt A is $f(A \cup \{u\}) - f(A)$

Definition 1.3 (Diminishing returns). f has diminishing returns if for $A \subseteq B$

$$f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B) \quad (2)$$

Proposition 1.1. f is submodular iff f has diminishing returns

Proof. \Rightarrow :

Let $A \subseteq B \subseteq N$ and $u \notin B$. Lets use submodularity property on $A \cup \{u\}$ and B :

$$f(A \cup \{u\}) + f(B) \geq f(A \cup \{u\} \cup B) + f((A \cup \{u\}) \cap B) = f(B \cup \{u\}) + f(A) \quad (3)$$

Thus

$$f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B) \quad (4)$$

□

\Leftarrow :

We'll proof by induction over $|A \cup B| - |A \cap B|$, i.e., size of symmetric difference.

Basis: $|A \cup B| - |A \cap B| = 0$, then $A = B$, and then submodular property is fulfilled.

Step: assume $|A \cup B| - |A \cap B| = k$. WLOG let $u \in A$ such that $u \notin B$.

$$f(A) + f(B) = f(A) - f(A \setminus \{u\}) + f(A \setminus \{u\}) + f(B) \geq \quad (5)$$

$$\geq f(A) - f(A \setminus \{u\}) + f(A \setminus \{u\} \cup B) + f(A \setminus \{u\} \cap B) \geq \quad (6)$$

$$\geq f(A \cup B) - f(A \cup B \setminus \{u\}) + f(A \cup B \setminus \{u\}) + f(A \cap B) = f(A \cup B) + f(A \cap B) \quad (7)$$

Definition 1.4 (Monotonous function). f is non-decreasing monotonous if $\forall A \subseteq B \subseteq N$, $f(A) \leq f(B)$.

Definition 1.5 (Symmetric function). f is symmetric if $\forall S \subseteq N$, $f(S) \leq f(N \setminus S)$.

Definition 1.6 (Normalized function). f is normalized if $f(\emptyset) = 0$.

Examples

Linear function $\forall n \in N$ exists weight w_n and

$$f(S) = \sum_{u \in S} w_u + b \quad (8)$$

Such f is submodular.

Budget additive function (clipped linear function) $\forall n \in N$ exists weight w_n and

$$f(S) = \min \left\{ \sum_{u \in S} w_u, b \right\} \quad (9)$$

Such f is submodular.

Coverage function Given set X and n subsets $S_1, S_2, \dots, S_n \subset X$ define

$$f(S) = \left| \bigcup_{i \in S} S_i \right| \quad (10)$$

This f is obviously submodular.

Graph cuts Let $G = (V, E)$ be a graph and $w : E \rightarrow \mathbb{R}^+$ weights of edges. Given a cut $S \subseteq V$ define $\delta(S)$ to be sum of weights of all edges going through the cut. $\delta : 2^V \rightarrow \mathbb{R}^+$ is submodular, normalized, and symmetric.

Rank function Let $v_1, \dots, v_n \in \mathbb{R}^d$ vectors, and

$$f(S) = \text{rank}(S) = \dim \text{span}(\{v_i | i \in S\}) \quad (11)$$

2 Submodular optimization

Given world N , submodular function $f : 2^N \rightarrow \mathbb{R}^+$, and a family of feasible solutions $\mathcal{I} \subseteq 2^N$

$$\max f(S) \quad (12)$$

$$\text{s.t. } S \in \mathcal{I} \quad (13)$$

Note Most of submodular functions (except for logarithm of determinant of submatrix) are nonnegative. We use the condition to have properly defined multiplicative approximation.

Note How f is given in input? Obviously, not as a list of values, since it's exponential in $|N|$. Thus we represent f with black box, and same applies for constraints. Usually, constraints are simple.

2.1 Examples of submodular optimization problems

Example f is submodular and there are no constraints. It generalizes MAX-CUT, MAX-DICUT

Example f is submodular and there is size constraint:

$$\max f(S) \quad (14)$$

$$\text{s.t. } |S| \leq k \quad (15)$$

. It generalizes MAX-K-COVER.

Submodular welfare

3 Maximization of the submodular function with cardinality constraints

$$\max f(S) \quad (16)$$

$$\text{s.t. } |S| \leq k \quad (17)$$

Algorithm 1 Nemhauser-Wolsey-Fisher

```
1: procedure GREEDY( $N$ )
2:    $A \leftarrow \emptyset$ 
3:   for  $i = 1$  to  $k$  do
4:     Let  $u_i \in N$  maximize  $f(A_{i-1} \cup \{u_i\}) - f(A_{i-1})$ 
5:      $A_i \leftarrow A_{i-1} \cup \{u_i\}$ 
6:   end for
7:   return  $A_k$ 
8: end procedure
```

Greedy algorithm If f is monotonic, greedy algorithm is an optimal approximating algorithm.

Lemma 3.1. For submodular $f : 2^N \rightarrow \mathbb{R}_+$,

$$f(A \cup B) - f(A) \leq \sum_{b_i \in B} f(A \cup \{b_i\}) - f(A) \quad (18)$$

Proof.

$$f(A \cup B) - f(A) = \sum_i f(A \cup \{b_1, \dots, b_{i-1}\} \cup \{b_i\}) - f(A \cup \{b_1, \dots, b_{i-1}\}) \leq \sum_i f(A \cup \{b_i\}) - f(A) \quad (19)$$

□

Proposition 3.2 (Nemhauser et al. [1978]). Algorithm 1 is $1 - \frac{1}{e}$ optimal.

Proof. For optimal set S^*

$$f(A_{i-1} \cup \{u_i\}) - f(A_{i-1}) \geq \max_{u \in S^*} \{f(A_{i-1} \cup \{u\}) - f(A_{i-1})\} \geq \frac{1}{k} \sum_{u \in S^*} [f(A_{i-1} \cup \{u\}) - f(A_{i-1})] \geq \quad (20)$$

$$\geq \frac{1}{k} \left(f(A_{i-1} \cup S^*) - f(A_{i-1}) \right) \geq \frac{1}{k} \left[f(S^*) - f(A_{i-1}) \right] \quad (21)$$

We got a recursion equation:

$$f(A_{i-1} \cup \{u_i\}) - f(A_{i-1}) \geq \frac{1}{k} \left[f(S^*) - f(A_{i-1}) \right] \quad (22)$$

We can solve the recursion and acquire

$$f(A_k) \geq \left(1 - \left(1 - \frac{1}{k} \right)^k \right) f(S^*) + \left(1 - \frac{1}{k} \right)^k f(A_0) \geq \left(1 - \frac{1}{e} \right) f(S^*) \quad (23)$$

□

Theorem 3.3 (Nemhauser and Wolsey [1978]). For all constant $\epsilon > 0$ each algorithm acquiring $1 - \frac{1}{e} + \epsilon$ requires exponential number of requests to value oracle.

Theorem 3.4 (Feige [1998]). For MAX-K-COVER all constant $\epsilon > 0$ each algorithm acquiring $1 - \frac{1}{e} + \epsilon$ requires exponential number of requests to value oracle unless $P = NP$.

Note Runtime of Algorithm 1 is $\mathcal{O}(nk)$. It is possible to acquire $\mathcal{O}(n \lg(\frac{1}{\epsilon}))$ runtime and $1 - \frac{1}{e} - \epsilon$ optimality by looking on some subset of N at each step instead of the whole set.

3.1 Non-monotonic functions

What happens if f is not monotonic? First of all, does greed algorithm work? Not only it is not optimal approximation, it can be as bad as $\frac{2}{N}$. However, it can be fixed. The idea is to randomize algorithm to prevent it from “bad” choices.

Algorithm 2

```

1: procedure RANDOMIZED GREEDY( $N$ )
2:    $A \leftarrow \emptyset$ 
3:   for  $i = 1$  to  $k$  do
4:      $M_i \leftarrow \arg \max_{B \subseteq N : |B| \leq k} \sum_{u \in B} f(A_{i-1} \cup \{u\}) - f(A_{i-1})$ 
5:      $A_i \leftarrow \begin{cases} A_{i-1} \cup \{u\} & \forall u \in M_i \text{ with } P = \frac{1}{k} \\ A_{i-1} & \text{with } P = 1 - \frac{|M_i|}{k} \end{cases}$ 
6:   end for
7:   return  $A_k$ 
8: end procedure

```

Randomized greedy algorithm

Theorem 3.5 (Buchbinder et al. [2014]). In monotonic case, Algorithm 2 is $1 - \frac{1}{e}$ optimal in expectation.

Proof. Take a look at i^{th} iteration and condition on previous iterations, denote a chosen element from M_i as u_i :

$$\mathbb{E} \left[f(A_{i-1} \cup \{u_i\}) - f(A_{i-1}) | A_{i-1} \right] = \frac{1}{k} \sum_{u_i \in M_i} f(A_{i-1} \cup \{u_i\}) - f(A_{i-1}) \geq \frac{1}{k} (f(A_{i-1} \cup S^*) - f(A_{i-1})) \geq \quad (24)$$

$$\geq \frac{1}{k} (f(S^*) - f(A_{i-1})) \geq \frac{1}{k} (f(S^*) - f(A_{i-1})) \quad (25)$$

If the inequality is right for any A_{i-1} it is right, from tower property, in expectation over A_{i-1} :

$$\mathbb{E} \left[f(A_{i-1} \cup \{u_i\}) - f(A_{i-1}) \right] \geq \frac{1}{k} (f(S^*) - \mathbb{E}[f(A_{i-1})]) \quad (26)$$

And thus we can once again solve the recurrence and acquire same result as in Proposition 3.2 on the previous page. \square

Lemma 3.6. Given set $B \subseteq N$ such that

$$\forall u \in N \quad P(u \in B) \leq p \quad (27)$$

then

$$\mathbb{E}[f(B)] \geq (1 - p)f(\emptyset) \quad (28)$$

Proof. WLoG $p(u_1 \in B) \geq p(u_2 \in B) \geq \dots \geq p(u_n \in B)$. Denote

$$X_i = \mathbb{1}_{u_i \in B} N_i = \bigcup_{j=1}^i u_j \quad (29)$$

We can then rewrite

$$f(B) = f(N_0) + \sum_{i=1}^n X_i \left(f(B \cap N_i) - f(B \cap N_{i-1}) \right) \quad (30)$$

$$\mathbb{E}[f(B)] = f(N_0) + \sum_{i=1}^n \mathbb{E} \left[X_i \left(f(B \cap N_i) - f(B \cap N_{i-1}) \right) \right] \geq \quad (31)$$

$$\geq f(N_0) + \sum_{i=1}^n \left(f(N_i) - f(N_{i-1}) \right) \mathbb{E}[X_i] = f(N_0) + \sum_{i=1}^n \left(f(N_i) - f(N_{i-1}) \right) p_i = \quad (32)$$

$$= f(N_0)(1 - p_1) + \sum_{i=1}^n f(N_i) \underbrace{(p_i - p_{i+1})}_{\leq 0} \geq f(N_0)(1 - p_1) \geq f(\emptyset)(1 - p) \quad (33)$$

\square

Lemma 3.7. Given set $A \subseteq N$ and set $B \subseteq N$ such that

$$\forall u \in N \quad P(u \in B) \leq p \quad (34)$$

$$\mathbb{E}[f(A \cup B)] \geq (1 - p)f(A) \quad (35)$$

Proof. Define

$$g_A(S) = f(A \cup S) \quad (36)$$

Obviously, g_A is also submodular (from diminishing returns). Then, from Lemma 3.6 on the preceding page

$$\mathbb{E}[f(A \cup B)] = \mathbb{E}[g_A(B)] \geq (1 - p)g_A(\emptyset) = (1 - p)f(A) \quad (37)$$

□

Theorem 3.8 (Buchbinder et al. [2014]). In non-monotonic case, Algorithm 2 on the previous page is $\frac{1}{e}$ optimal in expectation.

Proof. Similarly to monotonic case, take a look at i^{th} iteration and condition on previous iterations, denote a chosen element from M_i as u_i :

$$\mathbb{E}\left[f(A_{i-1} \cup \{u_i\}) - f(A_{i-1}) | A_{i-1}\right] = \frac{1}{k} \sum_{u_i \in M_i} f(A_{i-1} \cup \{u_i\}) - f(A_{i-1}) \geq \frac{1}{k}(f(A_{i-1} \cup S^*) - f(A_{i-1})) \quad (38)$$

Since

$$P(u \in A_{i-1}) \leq 1 - \left(1 - \frac{1}{k}\right)^{i-1} \quad (39)$$

from Lemma 3.7

$$\mathbb{E}[f(A_{i-1} \cup S^*)] \geq \left(1 - \frac{1}{k}\right)^{i-1} f(S^*) \quad (40)$$

Thus, taking expectation

$$\mathbb{E}\left[f(A_{i-1} \cup \{u_i\}) - f(A_{i-1})\right] \geq \frac{1}{k}(f(A_{i-1} \cup S^*) - f(A_{i-1})) \geq \frac{1}{k}\left[\left(1 - \frac{1}{k}\right)^{i-1} f(S^*) - \mathbb{E}[f(A_{i-1})]\right] \quad (41)$$

$$\mathbb{E}\left[f(A_i)\right] \geq \frac{1}{k}(f(A_{i-1} \cup S^*) - f(A_{i-1})) \geq \frac{1}{k}\left[\left(1 - \frac{1}{k}\right)^{i-1} f(S^*) - \mathbb{E}[f(A_{i-1})]\right] \quad (42)$$

Solving the recurrence we get

$$\mathbb{E}[f(A)] \geq \frac{i}{k}\left(1 - \frac{1}{k}\right)^{k-1} f(S^*) \geq \frac{1}{e}f(S^*) \quad (43)$$

i.e.,

$$\mathbb{E}[f(A_k)] \geq \left(1 - \frac{1}{k}\right)^{k-1} f(S^*) \geq \frac{1}{e}f(S^*) \quad (44)$$

□

Note Algorithm 2 on the previous page is not optimal. In addition, the upper bound of the best approximation is 0.49.

Runtime Runtime of Algorithm 2 on the preceding page is $\mathcal{O}(nk)$.

4 Maximization of the submodular function without constraints

$$\max f(S) \tag{45}$$

Examples

- MAX-CUT
- MAX-DIRECTED-CUT
- Max Facility Location
- MAX-SAT (with all literals in a clause having same sign).

Proposition 4.1 ([Feige et al., 2011]). Algorithm which choose random solution as following: $u \in S$ with probability $\frac{1}{2}$ independently, is $\frac{1}{4}$ approximation in expectation:

$$\mathbb{E}[f(S)] \geq \frac{1}{4} f(S^*) \tag{46}$$

Proposition 4.2 ([Feige et al., 2011]). If f is symmetric, the same algorithm is $\frac{1}{2}$ approximation in expectation:

$$\mathbb{E}[f(S)] \geq \frac{1}{2} f(S^*) \tag{47}$$

Proposition 4.3 ([Feige et al., 2011]). For any constant $\epsilon > 0$ it is impossible to acquire $(\frac{1}{2} + \epsilon)$ approximation in polynomial time, even in symmetric case.

Note that for $\bar{f}(S) = f(\bar{S})$, we can use the same oracle. So a "conjugate" algorithm would be start from N and drop elements from it.

Algorithm 3

```

1: procedure DOUBLE GREEDY( $N$ )
2:    $X \leftarrow \emptyset, Y \leftarrow N$ 
3:   for  $i = 1$  to  $n$  do
4:      $a_i = f(X_{i-1} \cup \{u_i\}) - f(X_i)$ 
5:      $b_i = f(Y_{i-1} \setminus \{u_i\}) - f(Y_i)$ 
6:     if  $a_i > b_i$  then
7:        $X_i \leftarrow X_{i-1} \cup \{u_i\}$ 
8:        $Y_i \leftarrow Y_{i-1}$ 
9:     else
10:       $X_i \leftarrow X_{i-1}$ 
11:       $Y_i \leftarrow Y_{i-1} \setminus \{u_i\}$ 
12:     end if
13:   end for
14:   return  $X_N$ 
15: end procedure

```

Algorithm 4

```

1: procedure RANDOMIZED DOUBLE GREEDY( $N$ )
2:    $X \leftarrow \emptyset, Y \leftarrow N$ 
3:   for  $i = 1$  to  $n$  do
4:      $a_i = \max \{0, f(X_{i-1} \cup \{u_i\}) - f(X_i)\}$ 
5:      $b_i = \max \{0, f(Y_{i-1} \setminus \{u_i\}) - f(Y_i)\}$ 
6:      $(X_i, Y_i) \leftarrow \begin{cases} (X_{i-1} \cup \{u_i\}, Y_i) & \text{with } P = \frac{a_i}{a_i + b_i} \\ (X_{i-1}, Y_{i-1} \setminus \{u_i\}) & \text{with } P = \frac{b_i}{a_i + b_i} \end{cases}$ 
7:   end for
8:   return  $X_N$ 
9: end procedure

```

Proposition 4.4. It's impossible that both $f(X_{i-1} \cup \{u_i\}) - f(X_{i-1}) < 0$ and $f(Y_{i-1} \setminus \{u_i\}) - f(Y_i) < 0$.

Proof. From diminishing returns:

$$f(X_{i-1} \cup \{u_i\}) - f(X_{i-1}) \geq f(Y_i) - f(Y_{i-1} \setminus \{u_i\}) \quad (48)$$

$$f(X_{i-1} \cup \{u_i\}) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1} \setminus \{u_i\}) \geq 0 \quad (49)$$

Thus at least one of $f(X_{i-1} \cup \{u_i\}) - f(X_{i-1})$ and $f(Y_{i-1} \setminus \{u_i\}) - f(Y_i)$ is greater than 0. \square

Lemma 4.5. Let S^* be an optimal solution and

$$S_i^* = S^* \cup X_i \cap Y_i \quad (50)$$

i.e., optimal solution to which we add everything Algorithm 3 added and drop everything it dropped.

For all i :

$$f(S_{i-1}^*) - f(S_i^*) \leq f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1}) \quad (51)$$

Lemma 4.6. Let S^* be an optimal solution and

$$S_i^* = S^* \cup X_i \cap Y_i \quad (52)$$

i.e., optimal solution to which we add everything Algorithm 4 added and drop everything it dropped.

For all i :

$$\mathbb{E}[f(S_{i-1}^*) - f(S_i^*)] \leq \frac{1}{2} \mathbb{E}[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})] \quad (53)$$

Proof. Take a look at i^{th} iteration and condition on previous iterations:

$$\mathbb{E}[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1}) \mid X_{i-1}, Y_{i-1}] = \quad (54)$$

$$= \frac{a_i}{a_i + b_i} \underbrace{(f(X_{i-1} \cup \{u_i\}) - f(X_{i-1}))}_{=a_i \text{ if } a_i \neq 0} + \frac{b_i}{a_i + b_i} \underbrace{(f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1}))}_{=b_i \text{ if } b_i \neq 0} = \frac{a_i^2 + b_i^2}{a_i + b_i} \quad (55)$$

Now divide into two cases: $u_i \in S^*$ and $u_i \notin S^*$.

- If $u_i \notin S^*$, in particular, $u_i \notin S_{i-1}^*$:

$$\mathbb{E}[f(S_{i-1}^*) - f(S_i^*)] = \frac{a_i}{a_i + b_i} (f(S_{i-1}^*) - f(S_{i-1}^* \cup \{u_i\})) \stackrel{S_{i-1}^* \subseteq Y_{i-1} \setminus \{u_i\}}{\leq} \quad (56)$$

$$\leq \frac{a_i}{a_i + b_i} (f(Y_{i-1}^* \setminus \{u_i\}) - f(Y_{i-1}^*)) \leq \frac{a_i b_i}{a_i + b_i} \quad (57)$$

- If $u_i \in S^*$, in particular, $u_i \in S_{i-1}^*$:

$$\mathbb{E}[f(S_{i-1}^*) - f(S_i^*)] = \frac{b_i}{a_i + b_i} (f(S_{i-1}^*) - f(S_{i-1}^* \setminus \{u_i\})) \stackrel{X_{i-1} \subseteq S_{i-1}^* \setminus \{u_i\}}{\leq} \quad (58)$$

$$\leq \frac{b_i}{a_i + b_i} (f(X_{i-1}^* \cup \{u_i\}) - f(X_{i-1}^*)) \leq \frac{a_i b_i}{a_i + b_i} \quad (59)$$

And since $a_i^2 - 2a_i b_i + b_i^2 = (a_i - b_i)^2 \geq 0$ (and by tower property), we get the required. \square

Theorem 4.7 ([Buchbinder et al., 2015]). Algorithm 4 on the previous page is $\frac{1}{2}$ approximation in expectation.

Proof. Denote

$$S_{alg} = S_n^* = X_n = Y_n \quad (60)$$

Then

$$\mathbb{E} \left[f(S_0^*) - f(S_n^*) \right] \leq \frac{1}{2} \mathbb{E} \left[f(X_n) - f(X_0) + f(Y_n) - f(Y_0) \right] \quad (61)$$

$$\mathbb{E} \left[f(S^*) - f(S_{alg}) \right] \leq \frac{1}{2} \mathbb{E} \left[2S_{alg} - f(X_0) - f(Y_0) \right] \stackrel{f(S) \geq 0}{\leq} \mathbb{E} \left[S_{alg} \right] \quad (62)$$

Thus

$$\mathbb{E}[S_{alg}] \geq \frac{1}{2} \mathbb{E}[f(S^*)] \quad (63)$$

□

Collary 4.7.1. Algorithm 3 on page 6 is $\frac{1}{3}$ approximation.

Note Algorithms 3 and 4 on page 6 run in $\mathcal{O}(N)$ time.

5 Knapsack constraints

Let each element of set have price c_i and budget B , then

$$\max f(S) \quad (64)$$

$$\text{s.t. } \sum_{i \in S} c_i \leq B \quad (65)$$

Algorithm 5

```

1: procedure DENSITY GREEDY( $N$ )
2:    $S \leftarrow \emptyset$ 
3:   while  $N \neq \emptyset$  do
4:      $x^* \leftarrow \arg \max \left\{ \frac{f(S \cup \{x\}) - f(S)}{c_i} \right\}$ 
5:     if  $c(S) + c_{x^*} \leq B$  then
6:        $S \rightarrow S \cup \{x^*\}$ 
7:     end if
8:      $N \leftarrow N \setminus \{x^*\}$ 
9:   end while
10:  return  $S$ 
11: end procedure

```

Note that this is generalization of cardinality constraint.

Algorithm 6

```

1: procedure OPTIMIZED DENSITY GREEDY( $N$ )
2:    $S_1 \leftarrow$  output of Algorithm 5
3:    $S_2 \leftarrow \left\{ \arg \max_{\substack{i \in N \\ c_i \leq B}} f(i) \right\}$ 
4:   return  $\arg \max_{S \in \{S_1, S_2\}} f(S)$ 
5: end procedure

```

Proposition 5.1 ([Khuller et al., 1999]). Algorithm 6 is $\frac{1}{2}(1 - \frac{1}{e})$ -optimal.

Proposition 5.2. Algorithm 6 is $(1 - \frac{1}{\sqrt{e}})$ -optimal.

Theorem 5.3 ([Khuller et al., 1999, Sviridenko, 2004]). If a set of l most dense items in optimal solution S^* , it is possible to get good approximation to the optimal solution.

Enumerating all sets of up to 3 most dense items in optimal solution S^* , we can acquire $1 - \frac{1}{e}$ -approximation of optimal solution. Since cardinality constraint is a particular case of knapsack constraint, this is best polynomial approximation.

References

- Niv Buchbinder, Moran Feldman, Joseph Seffi Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1433–1452. Society for Industrial and Applied Mathematics, 2014. (cited on pp. 4 and 5)
- Niv Buchbinder, Moran Feldman, Joseph Seffi, and Roy Schwartz. A tight linear time $(1/2)$ -approximation for unconstrained submodular maximization. *SIAM Journal on Computing*, 44(5):1384–1402, 2015. (cited on p. 7)
- Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998. (cited on p. 3)
- Uriel Feige, Vahab S Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011. (cited on p. 6)
- Samir Khuller, Anna Moss, and Joseph Seffi Naor. The budgeted maximum coverage problem. *Information processing letters*, 70(1):39–45, 1999. (cited on p. 8)
- George L Nemhauser and Laurence A Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3(3):177–188, 1978. (cited on p. 3)
- George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions–i. *Mathematical programming*, 14(1):265–294, 1978. (cited on p. 3)
- Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004. (cited on p. 8)