Non-monotone Submodular Maximization under Matroid and Knapsack Constraints [5]

Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, Maxim Svirdenko

Merkouris Papamichail†

†IGP in Algorithms, Logic and Discrete Mathematics
NKUA, NTUA

Approximation Algorithms,
Fall Semester 2021

Table of Contents

Table of Constents

Results

Matroid Constraints

$k \geq 1$ constraints	$\left(\frac{1}{k+2+\frac{1}{k}+\epsilon}\right)$
$k = 1$ constraint	$\left(\frac{1}{4+\epsilon}\right)$
symmetric	$\left(\frac{1}{k+2+\epsilon}\right)$
$k \geq 2$ partition constraints	$\left(\frac{1}{k+1+\frac{1}{k-1}+\epsilon}\right)$
monotone, $k \geq 2$ partition constraints	$\left(\frac{1}{k+\epsilon}\right)$

Basis Matroid Constraints

$k = 1$ constraint	$\left(\frac{1}{6+\epsilon}\right)$
symmetric	$\left(\frac{1}{3} - \epsilon\right)$
two disjoint bases	$\left(\frac{1}{6} - \epsilon\right)$

Knapsack Constraints

$k \geq 1$ constraints $\quad \left(\frac{1}{5} - \epsilon\right)$

Table of Constents

Definitions

Submodular Function

Let Ω be a finite set, and a set function $f : 2^{\Omega} \to \mathbb{R}$. We will be calling f *submodular*, if it satisfies the following condition: For every $X, Y \subseteq \Omega$, with $X \subseteq Y$ and every $x \in \Omega \setminus X$, we have,

$$f(X \cup x) - f(X) \geq f(Y \cup x) - f(Y).$$

An Example: Cuts in Graphs

Number of Edges in a Cut

Let $G = (V, E)$ be an undirected graph. We define the function $f : 2^V \to \mathbb{R}$ as,

$$f(S) = |(S, V \setminus S)|.$$

The number of edges in the cut $(S, V \setminus S)$.

1. $f(\cdot)$ is a submodular, non-negative function.
2. $f(\cdot)$ is *symmetric*, i.e.

$$f(S) = |(S, V \setminus S)| = |(V \setminus S, S)| = f(V \setminus S).$$

3. $f(\cdot)$ is *non-monotone*.

Optimization of a Submodular Function

Minimization

1. The *unconstrained* minimization problem is computable in (strongly) *polynomial* time.
2. The $\mathrm{Min\text{-}Cut}$ problem is a special case of sumbodular minimization.
3. Adding a simple constraint, such as a *cardinality lower bound*, makes the problem $\mathrm{NP\text{-}hard}$.

Maximization

1. The maximization problem is $\mathrm{NP\text{-}hard}$, even in the *unconstraint* setting.
2. The $\mathrm{Max\text{-}Cut}$ problem is a special case of sumbodular non-negative maximization.
3. If we allow f to take negative values, the problem is *inapproximable* [3].

Table of Constents

What is a Matroid?

1. A discrete structure, which encodes the notion of *independence*.
2. This structures, captures the notion of incrementing a partial solution I by making independent choices.
3. The discrete analog of *linear independence*.

Matroid Axioms

Let E be a finite set. Let, also, $\mathscr{I} \subseteq 2^E$ be a collection of subsets of E. We call the pair $\mathcal{M} = (E, \mathscr{I})$ a *matroid*, if $\mathscr{I}$ satisfies the following axioms.

(I1) $\varnothing \in \mathscr{I}$.

(I2) If $I \in \mathscr{I}$ and $I' \subseteq I$, then $I' \in \mathscr{I}$.

(I3) *(Augmentation Property)* If $I_1, I_2 \in \mathscr{I}$ and $|I_1| < |I_2|$, then there is a $i \in I_2 \setminus I_1$, such that $I_1 \cup i \in \mathscr{I}$.

Maximal Independent Sets

1. A maximal independent set of a matroid is called *basis*.
2. Resemble the idea of a basis of a vector space.
3. We denote the set of bases of a matroid $\mathcal{M}$ with $\mathscr{B}(\mathcal{M})$.
4. The bases of a matroid $\mathcal{M}$ are *equicardinal*, i.e. if $B_1, B_2 \in \mathscr{B}(\mathcal{M})$, then $|B_1| = |B_2|$.

Exchange Property

Let $\mathcal{M}$ be a matroid, and two bases $B_1, B_2 \in \mathscr{B}(\mathcal{M})$. For every $b_1 \in B_1$, there is a $b_2 \in B_2$, such that $(B_2 \setminus b_2) \cup b_1 \in \mathscr{B}(\mathcal{M})$.

Matroid Constraints

Problem 1: Submodular Maximization under Matroid Constraints

Let E be a finite set, and $f : 2^E \to \mathbb{R}_{\geq 0}$ a *non-negative*, *non-monotone*, *submodular* function. Let $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k$ a collection of matroids on the same ground set E. We are required to find some $S \subseteq E$, such that

$$f(S) = \max \left\{ S \in \bigcap_{i \in [k]} \mathscr{I}(\mathcal{M}_k) \right\}$$

A Local Search Approach

Algorithm 1: Approximate Local Search

1. **Initialize:** $S \leftarrow \varnothing$.
2. **Let:** $\mathcal{E} \geq 1$
3. **Repeat:**
    1. Delete operation. **If** there is an $e \in S$, such that $f(S \setminus e) > \mathcal{E} f(S)$, **then:** $S \leftarrow S \setminus e$.
    2. Augment operation. **If** there is a $d \in E \setminus S$, such that:
        1. $S \cup d$ is independent in *all* k matroids,
        2. $f(S \cup d) > \mathcal{E} f(S)$,

        then: $S \leftarrow S \cup d$.
    3. Exchange operation. **If** there is a $d \in E \setminus S$, and some $\{e_1, e_2, \ldots, e_\lambda\} \subseteq E$, such that:
        1. $(S \setminus \{e_1, e_2, \ldots, e_\lambda\}) \cup d)$ is independent in *all* k matroids,
        2. $f(S \setminus \{e_1, e_2, \ldots, e_\lambda\}) \cup d) > \mathcal{E} f(S)$,

        then: $S \leftarrow S \setminus \{e_1, e_2, \ldots, e_\lambda\}) \cup d$.

Algorithm 1: Regarding the Neighborhood (1)

Dislodge Relation

Let $\mathcal{M} = (E, \mathscr{I})$ be a matroid, and two independent sets $I, J \in \mathscr{I}$. We say that $j \in J$ dislodges $i \in I$ if $(I \setminus i) \cup j \in \mathscr{I}$. We will write $j \triangleright i$.

Proposition

Let $\mathcal{M} = (E, \mathscr{I})$ be a matroid, and two independent sets $I, J \in \mathscr{I}$. The following hold:

1. For every $j \in J$, either $I \cup j \in \mathscr{I}$, or there is some $i \in I$, such that $j \triangleright i$.
2. For every $i \in I$, there is *at most* one $j \in J$, such that $j \triangleright i$.

Algorithm 1: Regarding the Neighborhood (2)

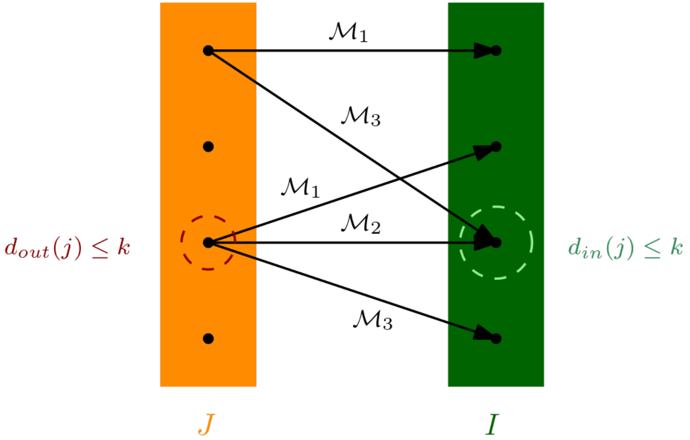

Figure 1: The representation of the *dislodge relation* as a bipartite graph $G = \left(E, \cup_{i \in [k]} \triangleright_i\right)$. Here, $k = 3$.

Algorithm 1: Use of Submodularity

Submodular Set-Function: Properties

Let $f : 2^E \to \mathbb{R}_{\geq 0}$ a non-negative, non-monotone submodular function. Let two sets, $S, T \supseteq \varnothing$, with $S \supseteq T$ and $C = \{c_1, c_2, \ldots, c_n\} \neq \varnothing$. The following hold:

1. $f(S \cup C) - f(S) \leq \sum_{i=1}^{n} f(T \cup c_i) - f(T)$.
2. $f(S \cup C) - f(S) = \sum_{i=1}^{n} f(S \cup \{c_1, \ldots, c_i\}) - f(S \cup \{c_1, \ldots, c_{i-1}\})$.

Lemma 1

For a *local optimum solution* S and any $C \in \cap_{i \in [k]} \mathscr{I}_i$, then

$$(k+1)f(S) \geq f(S \cup C) + k \cdot f(S \cap C)$$

Remarks

Algorithm 1: Approximate Local Search

1. **Initialize:** $S \leftarrow \varnothing$. **Let:** $\mathcal{E} \geq 1$
2. **Repeat:**
   1. Delete operation.

      If $\exists e \in S$, **s.t.:**
      1. $f(S \setminus e) > \mathcal{E} f(S)$,

      then: $S \leftarrow S \setminus e$.
   2. Augment operation.

      If $\exists d \in E \setminus S$, **s.t.:**
      1. $S \cup d \in \cap_{i \in [k]} \mathscr{I}_i$,
      2. $f(S \cup d) > \mathcal{E} f(S)$,

      then: $S \leftarrow S \cup d$.
   3. Exchange operation.

      If $\exists d \in E \setminus S$, and $\{e_1, e_2, \ldots, e_\lambda\} \subseteq E$,

      s.t.:
      1. $(S \setminus \{e_1, \ldots, e_\lambda\}) \cup d) \in \cap_{i \in [k]} \mathscr{I}_i$
      2. $f(S \setminus \{e_1, \ldots, e_\lambda\}) \cup d) > \mathcal{E} f(S)$,

      then: $S \leftarrow S \setminus \{e_1, \ldots, e_\lambda\}) \cup d$.

- The size of neighborhood is *at most* n^{k+1}.
- For *constant* k, each local step takes polynomial time.
- For $\mathcal{E} = 1$, we have an exponential time algorithm.
- We use $\mathcal{E} = \left(1 + \frac{\epsilon}{n^4}\right)$.
- This results to an *approximate* local search algorithm.
- Time Complexity: $n^{O(k)}$.

Conclusion

Algorithm 2: Greedy-Approximate-Local Search

1. **Initialize:** $E_1 \leftarrow E$
2. **For** $i \leftarrow 1$ **to** $k+1$ **:**
   1. $S_i \leftarrow \mathrm{Algorithm\text{-}1}\big(E_i,\ (1 + \frac{\epsilon}{n^4})\big)$.
   2. $E_{i+1} \leftarrow E_i \setminus S_i$.
3. **Return** $\max\{f(S_1), f(S_2), \ldots f(S_{k+1})\}$

Theorem 1

Algorithm-2 is a $\left(\frac{1}{(1+\epsilon)\left(k+2+\frac{1}{k}\right)}\right)$-approximation algorithm for maximizing a non-negative submodular function subject to any k matroid constraints, running in time $n^{O(k)}$.

Table of Constents

Knapsack Constraints

Problem 2: Submodular Maximization under Knapsack Constraints

Let E be a finite set, and $f : 2^E \to \mathbb{R}_{\geq 0}$ a non-negative, non-monotone, submodular function. Let $C_1, \ldots, C_k$ be the capacities of k knapsacks and $\mathbf{w}^1, \ldots, \mathbf{w}^k \in \mathbb{R}^n_{\geq 0}$ be the weights-vectors for each knapsack. We are required to find some $S \subseteq E$, such that,

$$f(S) = \max \left\{ \sum_{j \in S} w^i_j \leq C_i, \ \forall i \in [k] \right\}$$

- We assume $C_i = 1$, for all $i \in [k]$.
- Without loss of generality the sigleton $\{i\}$ is *feasible*, for all $i \in E$.

A Real Extension

Multilinear Extension

Let $\mathbf{x} \in [0,1]^n$ be a vector in $\mathbb{R}^n$. Then the *multilinear* extension of a submodular function $f\colon 2^E \to \mathbb{R}_{\geq 0}$ is a function $F\colon [0,1]^n \to \mathbb{R}_{\geq 0}$, such that,

$$F(\mathbf{x}) = \sum_{S \subseteq E} f(S) \prod_{i \in S} x_i \prod_{i \notin S} (1 - x_i)$$

1. F is *multilinear*, i.e. linear in each variable separately.
2. F is polynomial in variables $x_1, \ldots, x_n$.
3. F has *continuous derivatives* of any order.
4. Continuous submodularity. For all $i, j \in E$ $\frac{\partial^2}{\partial x_i \partial x_j} F \leq 0$ [6].

Continuous Submodularity

Claim

For any $\mathbf{a}, \mathbf{q}, \mathbf{d} \in [0,1]^n$ and $\mathbf{a} \leq \mathbf{q}$ coordinate-wise, we have

$$F(\mathbf{a} + \mathbf{d}) - F(\mathbf{a}) \geq F(\mathbf{q} + \mathbf{d}) - F(\mathbf{q}).$$

- Let $\mathbf{a}, \mathbf{q}, \mathbf{d}$ are characteristic vectors of the sets $S_{\mathbf{a}}, S_{\mathbf{q}}, S_{\mathbf{d}}$ respectively.
- Then, the above equation is equivalent of the 1st definition of a submodular function.
- For $\mathbf{x}, \mathbf{y} \in [0,1]^n$, let $\mathbf{x} \wedge \mathbf{y} := \min(x_i, x_j)$ and $\mathbf{x} \vee \mathbf{y} := \max(x_i, x_j)$.
- Then, we have the following as a corollary,

$$F(\mathbf{a} \vee \mathbf{d}) - F(\mathbf{a}) \geq F(\mathbf{q} \vee \mathbf{d}) - F(\mathbf{q}).$$

Fractional Relaxation

Problem 3: Fractional Relaxation of Problem 1

Let W be the matrix that has the weight vectors $\mathbf{w}^1, \ldots, \mathbf{w}^k \in \mathbb{R}^n_{\geq 0}$ as rows and $\mathrm{C} = [C_1 \ldots C_k]^t$ the column vector of the capacities. We are required to find some $\mathbf{x} \in [0,1]^n$, such that,

$$F(\mathbf{x}) = \max\{F(\mathbf{x}) \mid \mathrm{W}\mathbf{x} = \mathrm{C}\}$$

Solving the Fractional Relaxation: A Local Search Approach

Algorithm 3: Knapsack Approximate LS

1. **Let:** $\mathcal{G} = \left\{ p \cdot \zeta \mid p \in \mathbb{N},\ 0 \le p \le \frac{1}{\zeta} \right\}$.
2. **Let:** $\epsilon \ge 0$.
3. **Initialize:** $\mathbf{y} \leftarrow \arg\max\{F(\mathbf{e}^i) \mid i \in [n]\}$
4. **Repeat:**
   1. **Let:** $A, D \subseteq [n]$.
   2. `decrease` $\mathbf{y}(D)$ *in* $\mathcal{G}$,
   3. `increase` $\mathbf{y}(A)$ *in* $\mathcal{G}$,
   4. **resulting** to $\mathbf{y}'$,
   5. **while** $\mathrm{W}\mathbf{y}' = \mathrm{C}$
   6. **If** $F(\mathbf{y}') > (1+\epsilon)F(\mathbf{y})$: $\mathbf{y} \leftarrow \mathbf{y}'$
5. **Return:** $\mathbf{y}$.

Remarks

Algorithm 3: Knapsack Approximate LS

1. **Let:** $\mathcal{G} = \left\{ p \cdot \zeta \mid p \in \mathbb{N},\ 0 \leq p \leq \frac{1}{\zeta} \right\}$.
2. **Let:** $\epsilon \geq 0$.
3. **Initialize:** $\mathbf{y} \leftarrow \arg\max\{F(\mathbf{e}^i) \mid i \in [n]\}$
4. **Repeat:**
    1. **Let:** $A, D \subseteq [n]$.
    2. `decrease` $\mathbf{y}(D)$ *in* $\mathcal{G}$,
    3. `increase` $\mathbf{y}(A)$ *in* $\mathcal{G}$,
    4. **resulting** to $\mathbf{y}'$,
    5. **while** $\mathrm{W}\mathbf{y}' = \mathrm{C}$
    6. **If** $F(\mathbf{y}') > (1+\epsilon)F(\mathbf{y})$**:** $\mathbf{y} \leftarrow \mathbf{y}'$
5. **Return:** $\mathbf{y}$.

- The size of neighborhood is *at most* $n^{O(k)}$.
- For *constant* k, each local step takes polynomial time.
- We use $\zeta = \frac{1}{8n^4}$.
- This results to an *approximate* local search algorithm.
- Number of Iterations: $O\left(\frac{1}{\epsilon} \log n\right)$.

Fractional Relaxation: Approximation Guarantee

Lemma 2

For a local optimal solution $\mathbf{y}$ and *any* $\mathbf{x} \in [0,1]^n$ satisfying the knapsack constraints, we have

$$(2 + 2n + \epsilon) \cdot F(\mathbf{y}) \geq F(\mathbf{y} \wedge \mathbf{x}) + F(\mathbf{x} \vee \mathbf{y}) - \frac{1}{2n} F_{\max}$$

where $F_{\max} = \arg\max\{F(\mathbf{e}^i) \mid i \in [n]\}$.

Theorem 2

For any constant $\delta > 0$, there exists a $\left(\frac{1}{4} - \delta\right)$-approximation algorithm for *Problem 3*.

Solution to Problem 2: Rounding the Factional Solution

- We construct *2 solutions* & return the optimal.
- **Heavy Solution:**
  - For some δ an element $e \in E$ is *heavy* if $w(e) > \delta$.
  - The number of *heavy* elements in any feasible solution is *bounded* by $\frac{k}{\delta}$.
  - Enumerate all heavy solutions and obtain the optimal.
- **Light Solution:**
  - We obtain a feasible solution for Problem 3, from Algorithm 3.
  - We utilize a simple *randomized rounding* procedure.
  - This gives a $\left(\frac{1}{4} - \epsilon\right)$–approximation for Problem 3.
- Combining the above solutions we get a $\left(\frac{1}{5} - \epsilon\right)$–approximation for Problem 2.

Table of Constents

Conclusions & Progress Made

- A rich area of research in the recent years.
- Many other methods have been utilized including:
    - local search
    - continuous optimization
    - sampling
- The most of the papers consider a refinement of the problem, using a cardinality constraint.
- The current best approximation guarantee is 0.385 [2].
- The strongest inapproximability is 0.491 [4].
- Settling approximability of submodular maximization subject to a cardinality constraint remains an open problem [1].

Thank you for your time!

Table of Constents

Bibliography

- N. Alaluf, A. Ene, M. Feldman, H. L. Nguyen, and A. Suh, *Optimal streaming algorithms for submodular maximization with cardinality constraints*, arXiv preprint arXiv:1911.12959, (2019).
- N. Buchbinder and M. Feldman, *Constrained submodular maximization via a nonsymmetric technique*, Mathematics of Operations Research, 44 (2019), pp. 988–1005.
- U. Feige, V. S. Mirrokni, and J. Vondrak, *Maximizing non-monotone submodular functions*, SIAM Journal on Computing, 40 (2011), pp. 1133–1153.
- S. O. Gharan and J. Vondrák, *Submodular maximization by simulated annealing*, in Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms, SIAM, 2011, pp. 1098–1116.
- J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko, *Non-monotone submodular maximization under matroid and knapsack constraints*, in Proceedings of the forty-first annual ACM symposium on Theory of computing, 2009, pp. 323–332.
- J. Vondrák, *Optimal approximation for the submodular welfare problem in the value oracle model*, in Proceedings of the fortieth annual ACM symposium on Theory of computing, 2008, pp. 67–74.