Maximizing Non-monotone Submodular Set Functions Subject to Different Constraints: Combined Algorithms

Salman Fadaei[1]

Faculty of Engineering, University of Tehran, Tehran, Iran.

MohammadAmin Fazli [2]

Department of Computer Engineering, Sharif University of Technology, Tehran, Iran.

MohammadAli Safari [3]

Department of Computer Engineering, Sharif University of Technology, Tehran, Iran.

Abstract

We study the problem of maximizing constrained non-monotone submodular functions and provide approximation algorithms that improve existing algorithms in terms of either the approximation factor or simplicity. Our algorithms combine existing local search and greedy based algorithms. Different constraints that we study are (exact) cardinality and knapsack constraints. For the multiple-knapsack constraints we achieve a $(0.25 - \epsilon)$-factor algorithm.

We also show, as our main contribution, how to use the continuous greedy process for non-monotone functions and, as a result, obtain a 0.13-factor approximation algorithm for maximization over any down-monotone polytope. The continuous greedy process has been previously used for maximizing smooth monotone submodular function over a down-monotone polytope [CCPV08]. It appears that that our approach could be combined with pipage rounding or randomized swap rounding and provide approximation algorithms for various constrained maximization problems over non-monotone submodular functions. As an application of this technique we get a 0.13-approximation for maximizing non-monotone submodular functions subject to both matroid and multiple knapsacks.

Keywords:
non-monotone submodular set functions, approximation algorithms, continuous greedy process, matroid, knapsack, cardinality

1. Introduction

We study the problem of maximizing non-monotone non-negative submodular functions with respect to some constraints: packing polytope constraint, knapsack constraint and

[1]Email: fadaeis@ut.ac.ir
[2]fazli@ce.sharif.edu
[3]msafari@sharif.edu

cardinality constraints. Although these problems are extensively studied, we either propose simple algorithms or improve the existing approximation factors.

Definition 1. *A function $f\colon 2^X \rightarrow \mathbb{R}_+$ is called submodular if and only if $\forall A, B \subseteq X$, $f(A)+f(B) \geq f(A \cap B)+f(A \cup B)$. An alternative definition is that the marginal values of items should be non-increasing, i.e., $\forall A, B \subseteq X$, $A \subseteq B \subseteq X$ and $x \in X \setminus B$, $f_A(x) \geq f_B(x)$, where $f_A(x) = f(A \cup \{x\}) - f(A)$; $f_A(x)$ is called the marginal value of x with respect to A.*

The **Submodular Maximization Problem** *is a pair (f, Δ) where f is a submodular function and Δ is the search domain. Our aim is to find a set $A^* \in \Delta$ whose value, $f(A^*)$, is maximum. Our focus is on non-monotone submodular functions, i.e., are not necessarily monotone. That is, we do not require that $f(A) \leq f(B)$ for $A \subseteq B \subseteq X$.*

Definition 2. *A packing polytope is a polytope $P \subseteq [0,1]^X$ that is down-monotone: If $x, y \in [0,1]^X$ with $x \preceq y$ and $y \in P$, then $x \in P$. A polytope P is solvable if we can maximize linear functions over P in polynomial time [Dug09]. A* **Packing Polytope Constraint** *binds the search domain (Δ) to a packing polytope.*

Definition 3. *Suppose that we are given a ground set $X = \{a_1, a_2, ..., a_n\}$ and also a set of knapsacks $K = \{K_1, ..., K_k\}$. The capacity of each knapsack is one and the weight of a_i in K_j is given by some parameter $c_{i,j}$. A set $V \subseteq X$ is called packable if we can find a mapping $g : V \rightarrow K$ such that for all K_i, $\sum_{v \in V, g(v)=K_i} c_{v,i} \leq 1$.* **k-Knapsack Constraint** *forces us to bind search domain to packable subsets of X.*

Definition 4. *Consider a ground set X. In the* **Cardinality Constraint**, *$\Delta = \{S| \, |S| \leq k\}$ and in the* **Exact Cardinality Constraint**, *we have $\Delta = \{S| \, |S| = k\}$.*

Background.. Submodularity is the discrete analogous of convexity. Submodular set functions naturally arise in several different important problems including cuts in graphs [IFF01, GW94], rank functions of matroids [E70], and set covering problems [F98]. The problem of maximizing a submodular function is NP-hard as it generalizes many important problems such as Max Cut [FG95], Maximum Facility Location [B03, AS99], and the Quadratic Cost Partition Problem with non-negative edge weights [GG05].

The problem of maximizing non-monotone submodular functions, with or without some constraints, has been extensively studied in literature. In [FMV07] a 0.4-factor approximation algorithm was developed for maximizing general (non-negative, non-monotone) submodular functions subject to no constraints. The approximation factor was very recently improved to 0.41[OV10].

For the cases that we have constraints, Lee et al. [LMNS09], Vondrák [V09], and Gupta et al. [GRST10] have provided most outstanding approximation algorithms. In [LMNS09] a 0.2-approximation was developed for the problem subject to a constant number of knapsack constraints, followed by a 0.25-approximation for cardinality constraint and a 0.15-approximation the exact cardinality constraint. The latter two approximation factors were later improved in [V09] to 0.309, and 0.25, respectively. As a new way of tackling these problems, [GRST10] provides greedy algorithms that achieve the approximation factor of 0.17. This technique is more common for maximizing monotone submodular functions.

In a recent work, [V08] and [CCPV08] used the idea of multilinear extension of submodular functions and achieved good approximation algorithms for the problem of maximizing a

Constraint	[LMNS09]	[V09]	[GRST10]	Our Result	Claim
Exact Cardinality	0.15	0.25	0.17	0.25	Simpler
k-Knapsacks	0.2	-	-	0.25	Better ratio*
Packing Polytope	-	-	-	0.13	New ratio*

Table 1: Comparison of our results with the existing ones. $*$ - A 0.325-approximation algorithm has been achieved in [CVZ10] very recently.

monotone submodular function subject to a matroid. Using this technique, they first make the problem continuous, obtain a fractional solution using greedy algorithm, and finally round the fractional solution using randomized pipage rounding.

1.1. Our Results

We consider the problem subject to different constraints. Our results are summarized in Table 1. They are also compared with existing works. We obtain simple algorithms for cardinality constraint (maximum or exact), multiple knapsack constraints, and a new approximation factor for the packing polytope constraint.

1.2. Preliminaries

In this section we introduce the concepts and terms that we often use throughout this paper.

Multilinear Extension. For a submodular function $f\colon 2^X \to \mathbb{R}_+$, the multilinear extension of f can be defined as follows [CCPV07]: $F\colon [0,1]^X \to \mathbb{R}_+$ and

$$F(x) = \mathbf{E}[f(x)] = \sum_{S \subseteq X} f(S) \prod_{i \in S} x_i \prod_{i \in X \setminus S} (1 - x_i).$$

We use the concept of multilinear extension of a submodular function which is used frequently in recent works [CCPV07, CCPV08, KST09, LMNS09, V09].
A function $F : [0,1]^X \to R$ is smooth submodular if

- F has second partial derivatives everywhere.
- For any $i, j \in X$ (possibly equal), $\frac{\partial^2 F}{\partial x_i \partial x_j} \leq 0$ everywhere (submodularity).

The multilinear extension of every submodular function is a smooth submodular function [CCPV08]. The gradient of F is defined as $\nabla F = (\frac{\partial F}{\partial x_1} \dots \frac{\partial F}{\partial x_n})$.

Matroid. A matroid is a pair $\mathcal{M} = (X, \mathcal{I})$ where $\mathcal{I} \subseteq 2^X$ and

- $\forall B \in \mathcal{I}, A \subset B \Rightarrow A \in \mathcal{I}$
- $\forall A, B \in \mathcal{I}; |A| < |B| \Rightarrow \exists x \in B \setminus A; A + x \in \mathcal{I}$

If $\mathcal{I} = 2^X$, we call $\mathcal{M}$ a free matroid.

Matroid Polytopes. A matroid polytope is a solvable packing polytope with special properties. Given a matroid $\mathcal{M} = (X, \mathcal{I})$, we define the matroid polytope as

$$P(\mathcal{M}) = \{x \geq 0 \colon \forall S \subseteq X; \sum_{j \in S} x_j \leq r_{\mathcal{M}}(S)\}$$

where $r_{\mathcal{M}}(S) = max\{|I| \colon I \subseteq S; I \in \mathcal{I}\}$ is the rank function of matroid $\mathcal{M}$. This definition shows that the matroid polytope is a packing polytope.

Randomized Pipage Rounding. Pipage rounding was first introduced by Ageev and Sviridenko [AS04] for bipartite matching polytopes. Later, Calinescu et al. [CCPV07] adapted the technique to matroid polytopes. The adapted pipage rounding technique in [CCPV07] is a determinstic procedure. The randomized variant of the technique was later introduced and used in [CCPV08]. The randomized pipage rounding converts a fractional point in the matroid base polytope, $y \in B(\mathcal{M})$ into a random base $B \in \mathcal{M}$ such that $\mathbf{E}[f(B)] \geq F(y)$, where F is the multilinear extension of the submodular function f. Later, Vondrák [V09] extended the technique and showed that the starting point does not require to be inside the matroid base polytope and is enough to be inside a matroid polytope, $y \in P(\mathcal{M})$. Note that - as mentioned in [V09] - the submodular function does not need to be necessarily monotone.

It is proven in [CVZ09] that this rounding technique has some useful property called *negative correlation*, i.e., indicator vector variables X_i (which are the result of rounding x_i's) have expectation x_i and are negatively correlated.

2. Cardinality Constraint

In this section we propose very simple algorithms for some cardinality constraint problems whose approximation factor is either close or matches the best existing one, yet it is much simpler and easy to implement. Our algorithm is a simple combination of existing local search or greedy based algorithms. Our main tool is the following Theorem from Gupta et al. [GRST10].

Lemma 1. *([GRST10]). Given sets $C, S_1 \subseteq X$, let $C' = C \setminus S_1$, and $S_2 \subseteq X \setminus S_1$. Then $f(S_1 \cup C) + f(S_1 \cap C) + f(S_2 \cup C') \geq f(C)$.*

2.1. Cardinality Constraint

Theorem 1. *There is a 0.25-factor approximation algorithm for maximizing a non-monotone submodular function subject to a cardinality constraint.*

Proof. Lee et al. [LMNS09] use a local search approach and compute a local optimal set S_1 such that $2f(S_1) \geq f(S_1 \cup C) + f(S_1 \cap C)$ for any C with $|C| \leq k$. Gupta et al. [GRST10] propse a greedy based algorithm that computes a set S_2 such that for any C with $|C| \leq k$, $f(S_2) \geq 0.5 f(S_2 \cup C)$. Let C be the true optimum and $C' = C \setminus S_1$. Therefore,

$$2f(S_1) + 2f(S_2) \geq f(S_1 \cup C) + f(S_1 \cap C) + f(S_2 \cup C') \geq f(C) = OPT$$

Thus, an algorithm that returns the better of S_1 and S_2 has approximation factor 0.25 □

The approximation factor 0.25 is slightly smaller that the 0.309 factor of [V09], though our algorithm is simpler and straight forward to implement.

2.2. Exact Cardinality Constraint

An almost identical algorithms works for the exact cardinality problem. Let k be the cardinality number.

Theorem 2. *There is a* 0.25*-factor approximation algorithm for maximizing a non-monotone submodular function subject to an exact cardinality constraint.*

Proof. First, we use the local search algorithm of [LMNS09] and compute a set S_1 whose size is k and $2f(S_1) \geq f(S_1 \cup C) + f(S_1 \cap C)$ for any C with $|C| = |S_1| = k$. Next, we used the greedy algorithm of [GRST10] and compute a set $S_2 \subseteq X \backslash S_1$ of size k such that for any C' with $|C'| \leq k$, $f(S_2) \geq 0.5f(S_2 \cup C')$.

Again, the better of S_1 and S_2 gives an approximation factor 0.25. Here, we have assumed that $k \leq \frac{|X|}{2}$. If not, we can alternatively solve the problem for the derived submodular function $g(S) = f(X \setminus S)$ subject to cardinality constraint $k' = |X| - k$.

□

The approximation factor 0.25 matches that of [V09], though our algorithm is simpler and straight forward to implement.

3. Multiple Knapsack Constraints

Lee et al. [LMNS09] poropose an 0.2-factor approximation algorithm for the problem. They basically divide the elements into two sets of heavy and light objects and then solve the problem separately for each set and return the maximum of the two solutions.

We improve their result by considering both heavy and light elements together. Our algorithms finds a fractional solution and then integrates it by using pipage rounding. Our algorithms (shown as Algorithm 1 below) is based on the algorithm of Chekuri et al. [CVZ09] for maximizing monotone submodular functions subject to one matroid and multiple knapsack constraints. We have made some modifications to use it for non-monotone functions.

Input: Elements' weights $\{c_{i,j}\}$, parameter $0 < \epsilon < 1/(4k^2)$, and a non-monotone submodular function f

$D \leftarrow \emptyset$.

foreach *subset A of at most $1/\epsilon^4$ elements* **do**

0. Set $D \leftarrow A$ if $f(A) > f(D)$;

1. Redefine $C_j = 1 - \sum_{i \in A} c_{ij}$ for $1 \leq j \leq k$.;

2. Let B be the set of items $i \notin A$ such that either $f_A(i) > \epsilon^4 f(A)$ or $c_{ij} > k\epsilon^3 Cj$ for some j;

3. Let x^* be the fractional solution of the following problem:

$$max\{G(x) : x \in P(\mathcal{M}); \forall j \sum c_{ij} x_i \leq (1-\epsilon)C_j\} \quad (1)$$

where $\mathcal{M}$ is the free matroid over the ground set $X \setminus (A \cup B)$, and $G(x)$ is the multilinear extension of $g(S) = f_A(S)$, $S \subseteq X \setminus (A \cup B)$.;

4. Let R_A be the result of randomized pipage rounding applied to x^* with respect to the matroid polytope $P(\mathcal{M})$. Set $D \leftarrow A \cup R_A$ if $f(A \cup R_A) > f(D)$;

end

Return D.

Algorithm 1: Non-Monotone Maximization Subject to Multiple Knapsacks

. The following theorem shows how good our algorithms is.

Theorem 3. *Algorithm 1 returns a solution of value at least $(0.25 - 3\epsilon)OPT$ with hight probability.*

Proof. The proof follows the line of proofs of [CVZ09] with major changes to adapt it for non-monotone case. Let O be the optimal solution with $OPT = f(O)$. Assume $|O| \geq \frac{1}{\epsilon^4}$; otherwise, our algorithm finds the optimal solution in Line 0. Sort the elements of O by their decreasing marginal values, and let $A \subseteq O$ be the first $\frac{1}{\epsilon^4}$ elements. Consider the iteration in which this set A is chosen. Since A has $\frac{1}{\epsilon^4}$ elements, the marginal value of its last element and every element not in A is at most $\epsilon^4 f(A) \leq \epsilon^4 OPT$. So, throwing away elements whose marginal value is bigger than $\epsilon^4 f(A)$ does not hurt. We also throw away the set $B \subseteq N \setminus A$ of items whose size in some knapsack is more then $k\epsilon^3 Cj$. In $O \setminus A$, there can be at most $1/(k\epsilon^3)$ such items for each knapsack, i.e. $1/\epsilon^3$ items in total. Since their marginal values with respect to A are bounded by $\epsilon^4 OPT$, these items together have value $g(O \setminus B) = f_A(O \setminus B) \leq \epsilon OPT$. The set $O' = O \setminus (A \cup B)$ is still a feasible set for the maximization problem, and using submodularity, its value is Let $O' = O \setminus (A \cup B)$. We have

$$g(O') = g((O \setminus A) \setminus (O \setminus B)) \geq g(O \setminus A) - g(O \cap B) \geq OPT - f(A) - \epsilon OPT.$$

The indicator vector $(1-\epsilon)1_{O'}$ is a feasible solution for the problem 1 (specified at step 3 of algorithm 1). Using the concavity of $G(x)$ along the line from the origin to $1_{O'}$, we have $G((1-\epsilon)1_{O'}) \geq (1-\epsilon)g(O') \geq (1-2\epsilon)OPT - f(A)$. Using Theorem 4 of [LMNS09] we can compute in polynomial time a fractional solution x^* of value:

$$G(x^*) \geq \frac{1}{4} G((1-\epsilon)1_{O'}) \geq (\frac{1}{4} - 2\epsilon)OPT - f(A).$$

Finally, we apply randomized pipage rounding to x^* and call the resulting set R. By the construction of randomized pipage rounding, $g(R) \geq G(x^*)$. However, R might violate some of the knapsack constraints. Consider a fixed knapsack constraint, $\sum_{i \in S} c_{ij} \leq C_j$. Our fractional solution x^* satisfies $\sum c_{ij} x_i^* \leq (1-\epsilon)C_j$. Also we know that all sizes in the reduced instance are bounded by $c_{ij} \leq k\epsilon^3 C_j$. By scaling, $c'_{ij} = c_{ij}/(k\epsilon^3 C_j)$, we use Chernoff bound with $\mu = (1-\epsilon)/(k\epsilon^3)$:

$$Pr[\sum_{i \in R} c_{ij} > C_j] \leq Pr[\sum_{i \in R} c'_{ij} > (1+\epsilon)\mu] \leq e^{-\mu\epsilon^2/3} < e^{-1/4k\epsilon}.$$

Therefore, $Pr[\exists j : \sum_{j \in R} c_{ij} > C_j] \leq ke^{-1/4k\epsilon}$. For $\epsilon \leq 1/(4k^2)$ this probability is at most $ke^{-k} < 1$. Finally, we have a feasible solution of value $f(R) = f(A) + g(R) \geq f(A) + G(x^*) - \epsilon OPT \geq (\frac{1}{4} - 3\epsilon)OPT$.

□

4. Packing Polytope Constraint

In this section, we show that the continuous greedy process could be adapted for non-monotone submodular functions.As a result, we propose an algorithm for solving the optimization problems with packing polytope constraint. As an application of the technique, we then consider the problem of submodular maximization subject to both one matroid and multiple knapsacks constraints. Finally, we will show how to replace this continuous process with a polynomial time discrete process subject to an acceptable error.

4.1. Continuous greedy process for non-monotone functions

As stated in [CCPV08] the greedy process starts with $y(0) = \mathbf{0}$ and is increased over a unit time interval as follows:

$$\frac{dy}{dt} = v_{max}(y),$$

where $v_{max}(y) = argmax_{v \in P}(v.\nabla F(y))$.

We observe the following statement when F is not non-monotone.

Lemma 2. *$y(1) \in P$ and $F(y(1)) \geq (1-e^{-1})(F(x \vee y(1)) - F_{DMAX})$, where F is a smooth submodular function, $x \in P$ and $F_{DMAX} = \max_{0 \leq t \leq 1} F(y(1) - y(t))$.*

Proof. The proof is essentially similar to that of [CCPV08] with some modifications to adapt it for non-monotone functions. First of all, the trajectory for $t \in [0,1]$ is contained in P, since

$$y(t) = \int_0^t v_{max}(y(\tau))d\tau$$

is a convex linear combination of vectors in P. To prove the approximation guarantee, fix a point y. Consider a direction $v^* = (x \vee y) - y = (x - y) \vee 0$. This is a non-negative vector; since $v^* \leq x \in P$ and P is down-monotone, we also have $v^* \in P$. Consider the ray of direction v^* starting at y, and the function $F(y + \xi v^*)$, $\xi \geq 0$. The directional derivative of F along this ray is $\frac{dF}{d\xi} = v^*.\nabla F$. Since F is smooth submodular (that means, each entry

of ∇F, $\frac{\partial F}{\partial y_j}$ is non-increasing with respect to y_j) and v^* is nonnegative, $\frac{dF}{d\xi}$ is non-increasing too and $F(y+\xi v^*)$ is concave in ξ. By concavity:

$$F(y(1)+v^*)-F(y(t)) \leq F(y(t)+v^*)-F(y(t))+F(y(1)-y(t)) \leq v^*.\nabla F(y(t))+F_{DMAX}.$$

Since $v^* \in P$, and $v_{max}(y) \in P$ maximizes $v.\nabla F(y)$ over all vectors $v \in P$, we get

$$v_{max}(y).\nabla F(y) \geq v^*.\nabla F(y) \geq F(y(1)+v^*)-F_{DMAX}-F(y). \tag{2}$$

We now get back to the continuous process and analyse $F(y(t))$. By the chain rule and using (1), we get

$$\frac{dF}{dt}=\sum_j \frac{\partial F}{\partial y_j}\frac{dy_j}{dt}=v_{max}(y(t)).\nabla F(y(t)) \geq F(x \vee y(1))-F_{DMAX}-F(y(t)). \tag{3}$$

This means that $F(y(t))$ dominates the solution of the differential equation

$$\frac{d\phi}{dt}=F(x \vee y(1))-F_{DMAX}-\phi(t)$$

which means $\phi(t)=(1-e^{-t})(F(x \vee y(1))-F_{DMAX})$. This proves $F(y(t)) \geq (1-e^{-t})(F(x \vee y(1))-F_{DMAX})$.

□

4.2. Extending Smooth Local Search.

As our final tool for obtaining the main algorithm of this section, we propose an algorithm for the following problem: A submodular function, $f\colon 2^X \to \mathbb{R}_+$ is given. F is the multilinear extension of f. We have an uppper bound $\{u_i \in [0,1]\}_{i=1}^n$ for the variables. The problem is defind over the region $\mathcal{U} := \{0 \leq y_i \leq u_i \ \ \forall i \in X\}$:

$$\max\{F(y)\colon y \in \mathcal{U}\}$$

For this, we extend the 0.4-approximation algorithm (Smooth Local Search or SLS) of [FMV07] as follows. We call our algorithm FMV_Y.

We define a discrete set of values in $[0,1]$ as $\zeta=\{p.\delta\colon 0 \leq p \leq 1/\delta\}$ where $\delta=\frac{1}{8n^4}$ and p is integer. The result is a vector with the values selected from the discrete set ζ. We show that such a discretization does not substantially harm our solution, yet it reduces the running time.

First, we define a new function $g\colon 2^U \to \mathbb{R}_+$, $g(\cup_{i\in X}T_i)=F(\ldots,\frac{|T_i|}{s_i},\ldots)$ over the ground set U. U contains $s_i=\lfloor \frac{1}{\delta}u_i \rfloor$ copies of each element $i \in X$, and any subset $T \subseteq U$ is declared as $T=\cup_{i\in X}T_i$ where each T_i cosists of all copies of $i \in X$ from T. This function has been previously introduced and used in [LMNS09]. It is easy to verify that g is submodular [LMNS09]. Let B be the answer of running the the SLS algorithm to maximize g. B is a representation of some vector y and y is returned as the result.

Note that based on [FMV07]: $g(B) \geq 0.4g(A), \ \ \forall A \in U$, that means:

$$F(y) \geq 0.4F(z), \ \ \forall z \in \mathcal{U} \cap \zeta^n. \tag{4}$$

Claim 1. *For any $x \in \mathcal{U}$, $2.5F(y) \geq F(x) - \frac{f_{max}}{4n^2}$, where $f_{max} = \max\{f(i) \colon i \in X\}$.*

Proof. Let z be the point in $\zeta^n \cap \mathcal{U}$ that minimizes $\sum_{i=1}^{n}(x_i - z_i)$. Based on Claim 3 of [LMNS09], $F(z) \geq F(x) - \frac{f_{max}}{4n^2}$. Therefore, by also considering inequality (4) we conclude that $F(y) \geq 0.4(F(x) - \frac{f_{max}}{4n^2})$.

□

4.3. The Algorithm

Now we are ready to present our algorithm for maximizing a smooth submodular function over a packing polytope:

Input: A packing polytope P, a smooth submodular function F
1. $y_1 \longleftarrow$ the result of running the continuous greedy process.
2. $y'_1 \longleftarrow argmax_{0 \leq t \leq 1} F(y_1 - y(t))$.
3. $y_1^{max} \longleftarrow$ The result of running FMV_Y on with upper bound y_1.
4. $y^2 \longleftarrow$ the result of running the greedy process over the new polytope P' which is built by adding constraints $y_i \leq 1 - y_1$ for any $1 \leq i \leq n$ to P. Note that P' is a down-monotone polytope.
5. $y'_2 \longleftarrow argmax_{0 \leq t \leq 1} F(y_2 - y(t))$.
6. Return $argmax(F(y_1), F(y_2), F(y_{1max}), F(y'_1), F(y'_2))$.

Algorithm 2: Continuous greedy process for non-monotone functions

Theorem 4. *The above algorithm is a $\frac{2e-2}{13e-9}$-approximation algorithm for the problem of maximizing a smooth submodular function F, over a solvable packing polytope P.*

Proof. Suppose $x^* \in P$ is the true optimum with $F(x^*) = OPT$. by lemma 2, $F(y_1) \geq (1-e^{-1})(F(x^* \vee y_1) - F(y'_1))$. Also, we have $F(y_2) \geq (1-e^{-1})(F(x' \vee y_2) - F(y'_2))$, where $x' = x^* - (x^* \wedge y_1)$. Note that $x' \in P'$. Also, based on claim 1, $F(y_{1max}) \geq 0.4(F(x^* \wedge y_1) - \frac{f_{max}}{4n^2})$. Note that $x^* \wedge y_1 \preceq y_1$.
By adding up above inequalities we get

$$\begin{aligned} &\tfrac{e}{e-1}(F(y_1) + F(y_2)) + F(y'_1) + F(y'_2) + 2.5F(y_{1max}) \\ &\geq F(x^* \vee y_1) + F(x' \vee y_2) + F(x^* \wedge y_1) - \tfrac{f_{max}}{4n^2} \\ &\geq F(x^*) - \tfrac{f_{max}}{4n^2} = OPT - \tfrac{f_{max}}{4n^2}. \end{aligned}$$

Therefore the approximation factor of the algorithm is at least $\frac{2e-2}{13e-9}OPT$ □

Remark.. Observe that the step 3 of the algorithm could be done with other existing smaller approximations such as 0.25 or 0.3 instead of the 0.4.

Both one matroid and mutltiple knapsacks.. As a direct result of the above theorem, we propose the first approximation algorithm for maximizing a submodular function subject to both one matroid and mutltiple knapsack. This problem was solved (approximately) in [CVZ09] for monotone submodular functions.

Theorem 5. *There exists a $\frac{2e-2}{13e-9}$-approximation algorithm for the problem of maximizing a non-monotone submodular function f subject to one matroid and multiple knapsacks.*

Proof. It is enough to show that the intersection of the polytopes related to one matroid and multiple knapsacks is still a solvable packing polytope which is easy. Therefore, we can achieve a fractional solution by using Algorithm 2 besides the enumeration phase (as in Algorithm 1), and then using randomized pipage rounding the integral solution is achieved. □

4.4. Discrete Version

The main concern of this part is to propose a method for discretizing the continuos greedy process described in previous subsection 4.1. For this purpose, we can assume that the time interval is so small that the difference between the solution of differential inequality and the finite version of the algorithm is insignificant. However, since discretizing the time scale introduces some errors, here we describe the discrete version of the algorithm. The main result of this subsection is theorem 6 which corresponds to lemma 2. This theorem shows that our method could be discretized without producing any substantial error.

At first we mention the following useful lemma which may be the subject of independent interest:

Lemma 3. *For any $x, z \in \{0,1\}^X$ and any $y \in [0,1]^X$ such that $y \preceq z$ and let R denote a random set corresponding to y with elements sampled independently according to y_i then*

$$F(x \vee z) \leq F(y) + \max_{I \in \mathcal{I}} \sum_{j \in I} E[f_R(j)] + F_{DMAX}$$

where $F_{DMAX} = maxF(z - w)$, for any $w \in [0,1]^X$ such that $w \preceq z$.

Proof. Fix $A, B \in I$ such that $x = 1_A$ and $z = 1_B$. By submodularity $f(A \cup B) \leq f(AUR) + f(B - R)$ for any set $R \subseteq B$, Therefore, again by submodularity, $f(A \cup B) \leq f(R) + \sum_{j \in A} f_R(j) + f(B - R)$. By taking the expectation over a random subset R,

$$F(x \vee z) \leq F(y) + \sum_{j \in A} E[f_R(j)] + F(z - y) \leq F(y) + \max_{I \in \mathcal{I}} \sum_{j \in I} E[f_R(j)] + F_{DMAX}.$$

□

Now we present the discrete version of the algorithm. We assume ζ represents the set of different values which t (time) obtains: $\zeta = \{p.\delta \colon 0 \leq p \leq 1/\delta\}$. We can change the algorithm of [V08] for non-monotone submodular functions as follows:

Input: A matroid $\mathcal{M}$, and a non-monotone submodular function f
1. Let $\delta = 1/n^2$, where $n = |X|$, and start with $t = 0, y(0) = \mathbf{0}$.
2. Let $R(t)$ contains each j independently with probability $y_j(t)$. For each $j \in X$, estimate by sampling

$$w_j(t) = E[f(R(t) \cup \{j\}) - f(R(t))].$$

3. Let $I(t)$ be a maximum-weight independent set in M according to the weights $w_j(t)$. We can find this by the greedy algorithm for max-weight independent set with weights $w_j(t)$. Set

$$y(t + \delta) = y(t) + \delta \mathbf{1}_{I(t)}.$$

4. Increment $t := t + \delta$; if $t < 1$, go back to Step 2.
5. Let $y'_1 = argmax F(y(1) - y(t))$, for all $t \in \zeta$.
6. Return $y(1)$ and y'_1.

Algorithm 3: Discrete algorithm for simulating the continuous process

Remark.. Note that for fully discretizing Algorithm 2, one has to run Algorithm 3 at first, followed by running unconstrained submodular maximization (FMV_Y), and then running Algorithm 3 again with new parameters. The steps should be in accordance with the steps of Algorithm 2.

Before proceeding with the main theorem we mention some required lemmas. In the following lemmas we have some assumptions as follows: f is a non-monotone submodular function. $y(t+\delta)$ is the vector produced from $y(t)$ at time t: $y(t+\delta) = y(t)+\delta 1_{I(t)}$. $I(t)$ is the selected subset at time t by the Algorithm 3 at step 3. $R(t)$ and $R(t+\delta)$ are random subsets corresponding to $y(t)$ and $y(t+\delta)$, respectively. $D(t)$ is a random set that contains each item j independently with probability $\Delta_j(t) = y_j(t+\delta)-y_j(t)$, i.e. $\Delta(t) = y(t+\delta)-y(t) = \delta.1_{I(t)}$. so $D(t)$ is a random subset of $I(t)$ where each element appears independently with probability δ. ζ is the set values which t obtains during the algorithm execution.

Now we continue with the presentation of required lemmas and the proof details.

Lemma 4. *If there exists a* $0 \leq t \leq 1$ *such that* $E[f(R(t + \delta))] \leq E[f(R(t) \cup D(t))]$ *then* $E[f(R(t + \delta))] \geq E[f(R(t) \cup D(t))] - (\delta/n)OPT$.

Proof. Let $\theta = E[f(R(t + \delta))] - E[f(R(t) \cup D(t))]$. Let $\alpha(t) = R(t) \cup D(t)$ and $\beta(t) = R(t + \delta) \setminus \alpha(t)$. We observe that:

$$\begin{aligned} Pr[j \in \beta(t)] &= Pr[j \in R(t + \delta)] - Pr[j \in \alpha(t)] \\ &= y_j(t) + \Delta_j(t) - (1 - (1 - y_j(t))(1 - \Delta_j(t))) \\ &= \Delta_j(t).y_j(t). \end{aligned}$$

This means that, the elements of $\beta(t)$ are independently selected from $I(t)$ with the probability $\delta y_j(t)$. Therefore $\beta(t)$, intuitively, has few elements, maybe zero or one, but, for the completeness of the proof we consider the probability of all situations:

$$\begin{aligned} \theta &= \textstyle\sum_{j\in I(t)} Pr(\beta(t) = \{j\})\mathbf{E}[f_{\alpha(t)}(\{j\})] + \sum_{i,j\in I(t)} Pr(\beta(t) = \{i, j\})\mathbf{E}[f_{\alpha(t)}(\{i, j\})] + \dots \\ &= \textstyle\sum_{j\in I(t)} \delta y_j(t) \prod_{i\in I(t)\setminus\{j\}} (1 - \delta y_j(t))\mathbf{E}[f_{\alpha(t)}(\{j\})] + \dots \\ &\geq \delta^d \textstyle\prod_{j\in I(t)} y_j(t)(\sum_{j\in I(t)} \mathbf{E}[f_{\alpha(t)}(\{j\})] + \sum_{i,j\in I(t)} \mathbf{E}[f_{\alpha(t)}(\{i, j\})] + \dots) \\ &= \delta^d \textstyle\prod_{j\in I(t)} y_j(t).S \geq \delta^n \prod_{j\in I(t)} y_j(t).S \end{aligned}$$

Where $d = |I(t)|$. The first inequality occurs since $1 - \delta y_j(t) \geq \delta$.

Since $\theta \leq 0$, therefore

$$\begin{aligned} |\theta| &\leq \delta^n \textstyle\prod_{j \in I(t)} y_j(t).|S| \leq \delta^n.|S| \\ &\leq \delta^n(\textstyle\sum_{j \in I(t)} |\mathbf{E}[f_{\alpha(t)}(\{j\})]| + \sum_{i,j \in I(t)} |\mathbf{E}[f_{\alpha(t)}(\{i,j\})]| + \ldots) \\ &\leq \delta^n \tbinom{d}{1}.OPT + \tbinom{d}{2}.OPT + \ldots \\ &\leq \delta^n.2^d.OPT \leq \delta^n.2^n.OPT. \\ &\leq (\delta/n)OPT \end{aligned}$$

Again since $\theta \leq 0$ we have $\theta \geq -(\delta/n)OPT$, and therefore the lemma implies.

□

Lemma 5. *$\gamma \geq -(\delta/n)OPT$, where*

$$\begin{aligned} \gamma = \; & \textstyle\sum_{i,j \in I(t)} Pr(D(t) = \{i,j\}) \boldsymbol{E}[f_{R(t)}(\{i,j\})] + \\ & \textstyle\sum_{i,j,k \in I(t)} Pr(D(t) = \{i,j,k\}) \boldsymbol{E}[f_{R(t)}(\{i,j,k\})] + \ldots + \\ & \textstyle\sum_{I(t) = I(t)} Pr(D(t) = I(t)) \boldsymbol{E}[f_{R(t)}(I(t))]. \end{aligned}$$

Proof. If $\gamma \geq 0$ then lemma implies. Therefore, we suppose $\gamma < 0$. Let $|I(t)| = d$.

$$\begin{aligned} \gamma &= \textstyle\sum_{i,j \in I(t)} \delta^2(1-\delta)^{d-2} \mathbf{E}[f_{R(t)}(\{i,j\})] + \sum_{i,j,k \in I(t)} \delta^3(1-\delta)^{d-3} \mathbf{E}[f_{R(t)}(\{i,j,k\})] + \ldots \\ &\geq \delta^d(\textstyle\sum_{i,j \in I(t)} \mathbf{E}[f_{R(t)}(\{i,j\})] + \sum_{i,j,k \in I(t)} \mathbf{E}[f_{R(t)}(\{i,j,k\})] + \ldots) \\ &= \delta^d S \geq \delta^n S. \end{aligned}$$

The first inequality occurs since $1 - \delta \geq \delta$. We consider the value of S:

$$\begin{aligned} |S| &\leq (\textstyle\sum_{i,j \in I(t)} |\mathbf{E}[f_{R(t)}(\{i,j\})]| + \sum_{i,j,k \in I(t)} |\mathbf{E}[f_{R(t)}(\{i,j,k\})]| + \ldots) \\ &\leq \tbinom{d}{2}.OPT + \tbinom{d}{3}.OPT + \ldots \\ &\leq 2^d.OPT \leq 2^n.OPT. \end{aligned}$$

Since $\gamma \leq 0$, therefore $S \leq 0$ and $S \geq -2^n.OPT$ therefore, $\gamma \geq -2^n.\delta^n.OPT$.
Since, $2^n.\delta^n \leq \delta/n$, the lemma implies.

□

Lemma 6. *If $E[f(R(t+\delta))] \geq E[f(R(t) \cup D(t))] - \beta$ then $VAL - F(y(t+\delta)) \leq (1-\delta)(VAL - F(y(t)))$, where $VAL = F(x \vee y(1)) - F_{DMAX} - 3/nOPT - \beta/\delta$, and $F_{DMAX} = maxF(y(1) - y(t))$, for all $t \in \zeta$.*

Proof.

$$\begin{aligned} F(y(t+\delta)) - F(y(t)) &= E[f(R(t+\delta))] - E[f(R(t))] \\ &\geq E[f(R(t) \cup D(t))] - E[f(R(t))] - \beta \\ &= \textstyle\sum_{j \in I(t)} Pr(D(t) = \{j\}) \mathbf{E}[f_{R(t)}(\{j\})] - \beta + \gamma \\ &= \textstyle\sum_{j \in I(t)} \delta(1-\delta)^{|I(t)|-1} \mathbf{E}[f_{R(t)}(\{j\})] - \beta + \gamma \\ &\geq \delta(1 - n\delta) \textstyle\sum_{j \in I(t)} \mathbf{E}[f_{R(t)}(\{j\})] - \beta + \gamma \end{aligned}$$

where

$$\begin{aligned} \gamma = \; & \textstyle\sum_{i,j \in I(t)} Pr(D(t) = \{i,j\}) \mathbf{E}[f_{R(t)}(\{i,j\})] + \\ & \textstyle\sum_{i,j,k \in I(t)} Pr(D(t) = \{i,j,k\}) \mathbf{E}[f_{R(t)}(\{i,j,k\})] + \ldots + \\ & \textstyle\sum_{I(t) = I(t)} Pr(D(t) = I(t)) \mathbf{E}[f_{R(t)}(I(t))]. \end{aligned}$$

by Lemma 5, $\gamma \geq -(\delta/n)OPT$. Therefore,

$$F(y(t+\delta)) - F(y(t)) \geq \delta(1-n\delta) \sum_{j \in I(t)} \mathbf{E}[f_{R(t)}(\{j\})] - \beta - (\delta/n)OPT.$$

We incur an error of at most OPT/n in computation of the maximum-weight independent set. Then

$$\begin{aligned} F(y(t+\delta)) - F(y(t)) &\geq \delta(1-n\delta)(\max_{I \in \mathcal{I}} \textstyle\sum_{j \in I} E[f_R(j)] - OPT/n) - \beta - (\delta/n)OPT. \\ &\geq \delta(VAL - F(y(t))). \end{aligned}$$

The second inequlity is achieved by using lemma 3, $\delta = 1/n^2$, setting $VAL = F(x \vee y(1)) - F_{DMAX} - 3/nOPT - \beta/\delta$, where $F_{DMAX} = maxF(y(1) - y(t))$, for all $t \in \zeta$, and by using $F(x \vee y(1)) - F_{DMAX} \leq OPT$.
From here we have $VAL - F(y(t+\delta)) \leq (1-\delta)(VAL - F(y(t)))$.

□

The main result of this part is to show that discrete Algorithm 3 has the same result as the continuous greedy process described in subsection 4.1 (with a small amount of error). For getting better intuition you can compare Theorem 6 by Lemma 2.

Theorem 6. *$y(1), y'_1 \in P$ and $F(y(1)) \geq (1-e^{-1})(F(x \vee y(1)) - F(y'_1)) - o(1)OPT$, where $x \in P$.*

Proof. The proof is based on [V08]. They use the monotonocity of f, but since our function is not necessarily monotone we adapt it for our case. We estimate the growth of $F(y(t))$ at each step of the algorithm by starting from $F(y(0)) = 0$. We suppose $R(t)$ is a random set that corresponds to $y(t)$. Also, we assume $D(t)$ as a random set that contains each item j independently with probability $\Delta_j(t) = y_j(t+\delta) - y_j(t)$. The growth of $F(y(t))$ at time t is:

$$F(y(t+\delta)) - F(y(t)) = E[f(R(t+\delta))] - E[f(R(t))]$$

We compare $E[f(R(t+\delta))]$ with $E[f(R(t) \cup D(t))]$. When f is monotone $E[f(R(t+\delta))] \geq E[f(R(t) \cup D(t))]$ [V08]. But, here the function is not necessarily monotone. However, as shown by lemma 4 we have:

$$E[f(R(t+\delta))] \geq E[f(R(t) \cup D(t))] - (\delta/n)OPT. \tag{5}$$

Inequlity (5) shows that if $E[f(R(t+\delta))]$ is less than $E[f(R(t) \cup D(t))]$ then the difference will be a very small number. Therefore, we have:

$$E[f(R(t+\delta))] - E[f(R(t))] \geq E[f(R(t) \cup D(t))] - E[f(R(t))] - (\delta/n)OPT.$$

Then by lemma 6 we have $VAL - F(y(t+\delta)) \leq (1-\delta)(VAL - F(y(t))$, where $VAL = F(x \vee y(1)) - F(y'_1) - (4/n)OPT$.
By induction, $VAL - F(y(k\delta)) \leq (1-\delta)^k VAL$. For $k = 1/\delta$, we get

$$VAL - F(y(1)) \leq (1-\delta)^{1/\delta} VAL \leq (1/e)VAL.$$

Therefore, $F(y(1)) \geq (1-1/e)VAL \geq (1-1/e)(F(x \vee y(1)) - F(y'_1)) - o(1)OPT$. □

Acknowledgement

The Authors are grateful to Dr. Vahab Mirrokni for his many helps in all the steps of the preparation of this paper. The first author is also thankful to Dr. Ali moeini (his M.Sc. advisor), Dr. Dara Moazzami, and Jasem Fadaei for their help and advice.

References

[AS99] A. A. Ageev and M. I. Sviridenko, An 0.828-approximation algorithm for the uncapacitated facility location problem. Discrete Appl. Math., 93, 149156, 1999.

[AS04] A. Ageev and M. Sviridenko. Pipage rounding: a new method of constructing algorithms with proven performance guarantee. J. of Combinatorial Optimization, 8:307328, 2004.

[B03] S. Benati. An improved Branch & Bound method for the uncapacitated competitive location problem. Annals of Operations Research 122,no. 1, 43-58, 2003.

[CCPV07] G. Calinescu, C. Chekuri, M. Pál and J. Vondrák. Maximizing a monotone submodular function under a matroid constraint. Proc. of 12th IPCO, 182196, 2007.

[CCPV08] G. Calinescu, C. Chekuri, M. Pál and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint. To appear in SIAM Journal on Computing, special issue for STOC 2008.

[CKP00] A. Caprara, H. Kellerer and U. Pferschy. Approximation Algorithms for Knapsack Problems with Cardinality Constraints. European J. Oper. Res. 123, 333-345, 2000.

[CVZ09] C. Chekuri, J. Vondrák, R. Zenklusen. Dependent Randomized Rounding for Matroid Polytopes and Applications, 2009.

[CVZ10] C. Chekuri, J. Vondrák, and R. Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. Manuscript, 2010.

[Dug09] S. Dughmi. Submodular Functions: Extensions, Distributions, and Algorithms A Survey. http://arxiv.org/abs/0912.0322, 2009.

[E70] J. Edmonds. Matroids, submodular functions, and certain polyhedra. Journal of Combinatorial Structures and Their Applications, 69-87, 1970.

[F98] U. Feige. A threshold of ln n for approximating set cover. Journal of the ACM, 45(4):634652, 1998.

[FG95] U. Feige, and M.X. Goemans. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT Proceedings of the Third Israel Symposium on Theory of Computing and Systems 388, 1995.

[FMV07] U. Feige, V.S.Mirrokni, and J. Vondrák. Maximizing non-monotone submodular functions. In FOCS, 2007.

[GJ79] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Co., 1979.

[GG05] B. Goldengorin and D. Ghosh. The multilevel search algorithm for the maximization of submodular functions applied to the quadratic cost partition problem. Journal of Global Optimization 32 (1), 6582, 2005.

[GRST10] A. Gupta, A. Roth, G. Schoenebeck, K. Talwar. Constrained Non-Monotone Submodular Maximization: Offline and Secretary Algorithms, 2010.

[GW94] M. Goemans and P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. Journal of the ACM (JACM) 42(4), 1115-1145, 1994.

[HMS08] J. Hartline, V. Mirrokni, M. Sundararajan. Optimal marketing strategies over social networks. Proceeding of the 17th international conference on World Wide Web, 189-198, 2008.

[IFF01] S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions Journal of the ACM (JACM) 48(4), 777, 2001.

[KS10] A. Kulik, H. Shachnai. Improved Approximations for Non-monotone Submodular Maximization with Knapsack Constraints.
http://www.cs.technion.ac.il/∼hadas/PUB/non_monotone_submodular.pdf, 2010.

[KST09] A. Kulik, H. Shachnai and T. Tamir. Maximizing submodular set functions subject to multiple linear constraints. Proc. of ACM-SIAM SODA, 545554, 2009.

[LMNS09] J. Lee, V. Mirrokni, V. Nagarajan, and M. Sviridenko. Maximizing non-monotone submodular functions under matroid and knapsack constraints. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC), 2009.

[OV10] S. Oveis Gharan, J. Vondrák. Submodular Maximization by Simulated Annealing. http://arxiv.org/abs/1007.1632, 2010.

[S04] M. Sviridenko. A note on maximizing a submodular set function subject to knapsack constraint. Operations Research Letters 32, 4143, 2004.

[Sch03] A. Schrijver. Combinatorial optimization - polyhedra and efficiency. Springer, 2003.

[SU07] A. Schulz, and N. Uhan. Encouraging cooperation in sharing supermodular costs Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 271-285, 2007.

[V08] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In STOC, pages 67-74, 2008.

[V09] J. Vondrák. Symmetry and approximability of submodular maximization problems. In Proc. of IEEE FOCS, 251270, 2009.