Lower Bounds on the Discrete Newton's Algorithm for the Submodular Line Search Problem

SPUR+ Paper, Summer 2022

Evgeniya Artemova and Christina Yu
Mentor: Yuchong Pan
Project suggested by Michel Goemans

August 3, 2022

Abstract

Submodular functions arise naturally from the principle of diminishing marginal returns in economics. We are interested in the line search problem in the submodular polytope of a submodular function, which asks for the maximum step length within the polytope in a given search direction from a given starting point. This problem can be solved by the discrete Newton's algorithm. While linear and quadratic upper bounds are known on the number of iterations of the algorithm in the cases where the search direction is nonnegative and arbitrary, respectively, we have little knowledge on whether there are matching lower bounds in these two cases. In this paper, we provide several initial attempts towards lower bound constructions in these problems.

1 Introduction

In this section, we first introduce submodular functions, and define a line search problem in a polytope associated with a submodular function. Then, we present an iterative algorithm called the *discrete Newton's algorithm* which solves this line search problem.

1.1 Submodular Functions

Submodularity is an important concept in combinatorial optimization, and has attracted increasing interest from other relevant fields such as machine learning and game theory. Informally, submodularity characterizes the *diminishing returns property*, i.e., the property that the *marginal benefit* of adding an additional element to a set decreases as the input set expands. Formally, for all $A \subseteq B \subsetneq V$ and $i \in V \setminus B$,

$$f(A \cup \{i\}) - f(A) \geq f(B \cup \{i\}) - f(B).$$

Equivalently, we can define a submodular function by the *uncrossing operation*. In other words, a set function $f : 2^V \to \mathbb{R}$ is submodular if and only if for all $A, B \subseteq V$,

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B).$$

We are sometimes also interested in *supermodular* functions; we say that a set function f is *supermodular* if $-f$ is submodular. In addition, we say that a set function f is *modular* if f is both submodular and supermodular. It can be easily seen that a set function f on some finite ground set V is modular if and only if it is of the form $f(S) = \sum_{j \in S} b_j$ for some vector $b \in \mathbb{R}^V$, so modular functions are also called *linear* functions. Many classic combinatorial optimization problems have modular objective functions, such as the cardinality function.

Submodularity captures many combinatorial structures. We provide two examples in this section for the reader's reference. Firstly, if $f(S)$ is the number of connected components in the subgraph induced by edges S in an undirected graph G, then f is supermodular (and hence $-f$ is submodular). Secondly, given a matrix A, if $g(S)$ is the rank (i.e., the maximal number of independent columns) of the submatrix induced by a subset S of the columns, then g is submodular. Indeed, the notion of independence in a matrix can be generalized to that in a matroid. For conciseness, we avoid defining matroids in this short paper, but note that the rank function of any matroid is submodular. Matroid rank functions form an important subclass of submodular functions, and the connection between submodular functions and matroids is deep.

Submodularity gives us some sort of "discrete smoothness." On one hand, submodular functions play a similar role as convex functions in continuous optimization. An argument for this case is that convex functions and submodular functions are both easy to minimize but hard to maximize. A polynomial time exact algorithm is known to solve unconstrained submodular minimization, but we only have a 1/2-approximation for maximization. In addition, any set function is submodular if and only if its *Lovász extension* is convex. On the other hand, surprisingly, submodular functions share several properties of concave functions. An argument for this case is that the marginal benefit $f(A \cup \{i\}) - f(A)$ can be viewed as a discrete derivative $\partial_i f(x) = f(x + e_i) - f(x)$, and the diminishing returns property seems to require these discrete derivatives to be non-increasing, analagous to the definition of concavity.

Due to the limit of this paper, we refer the interested reader to the seminal survey of Lovász [5] for other examples, applications and properties of submodular functions, and their connections to convex and concave functions, respectively.

1.2 Submodular Line Search and Discrete Newton's Algorithm

Polyhedral combinatorics study polyhedra or polytopes associated with discrete sets that arise from combinatorial optimization problems, such as matchings and stable sets. A common theme in the research of submodular functions is to study polyhedra or polytopes associated with a submodular function. We define the *submodular polytope*, or *extended polymatroid*, of a submodular function $f : 2^V \to \mathbb{R}$ on some finite ground set V to be

$$P(f) = \left\{ x \in \mathbb{R}^V \mid x(S) \leq f(S) \; \forall S \subseteq V \right\},$$

where we denote $x(S) = \sum_{j \in S} x_j$ for any vector $x \in \mathbb{R}^V$ and $S \subseteq V$.

A line search problem asks for the maximum step length within a polytope in a given search direction from a given starting point. An application of line search is to implement the algorithmic version of Carathéodory's theorem in convex geometry, which we briefly explain below. The classic Carathéodory's theorem states the following:

Theorem 1 (Carathéodory's theorem)**.** *Any point $x \in \mathbb{R}^d$ lying in* conv P *for some finite set P with* conv P *full-dimensional can be written as the convex combination of at most $d+1$ points in P.*

For completeness, we give a proof of Carathéodory's theorem, which implies that line search can be used to find such a convex combination. This proof can be found in [7].

Proof. We proceed by induction on d. For the base case $d = 1$, any finite set $P \subseteq \mathbb{R}^1$ with $\dim(\text{conv } P) = 1$ has at least two vertices, so the theorem follows.

Let $d \in \mathbb{N}$. Assume that the theorem holds for all $d' \in \mathbb{N}$ with $d' \leq d$. Let $P \subseteq \mathbb{R}^{d+1}$ be finite and such that $\dim(\text{conv } P) = d + 1$. Let $x \in \text{conv } P$. Let v_1 be a vertex of conv P. WLOG, assume that x lies in the interior of P; otherwise, x lies on a face of P and we are done by the inductive hypothesis. Perform a line search in conv P from v_1 in the search direction $x - v_1$. Let x' be the intersection of the line and conv P. Since $x \in \text{conv } P$, then $x' = v_1 + \delta(x - v_1)$ for some $\delta \geq 1$. Let $\lambda_1 \in [0, 1]$ be such that $x = \lambda_1 v_1 + (1 - \lambda_1)x'$.

Hence, x' lies on a face F of P such that $F \neq \text{conv } P$, which is a polytope of dimension at most d. By induction, x' can be written as a convex combination of at most $d+1$ vertices of F. Since x is a convex combination of v_1 and x', then x can be written as a combination of at most $d + 2$ vertices of conv P, completing the proof. □

In particular, we are interested in the following line search problem in the submodular polytope of a submodular function:

Problem 2 (Submodular line search problem)**.** *Given a submodular function $f : 2^V \to \mathbb{R}$ on some finite ground set V, a vector $x_0 \in P(f)$ and a vector $a \in \mathbb{R}$, find the largest δ such that $x_0 + \delta a \in P(f)$.*

This can be interpreted as the polytope giving you a boundary, and within this boundary you have an initial point x_0. Then you are given some direction a, and you are trying to work out how far you can go in that direction without reaching a boundary.

Without loss of generality, we can assume that f is non-negative. To see this, let $f' : V \to \mathbb{R}$ be defined by $f'(S) = f(S) - x_0(S)$, and then finding the largest δ such that $\delta a \in P(f')$. We also notice that since $x_0 \in P(f)$, we have that for all $S \subseteq V$, $x_0(s) \leq f(S)$ if and only if $f(S) - x_0(S) \geq 0$ and so $0 \in P(f')$. This then implies $f'(S) \geq 0$ for all subsets $S \subseteq V$. Thus we can consider the following equivalent problem

$$\delta^* = \max \left\{ \delta \mid \min_{S \subseteq V} f(S) - \delta a(S) \geq 0 \right\}. \tag{1}$$

This problem can be solved by the discrete Newton's algorithm by using the cutting plane method ([3]). The discrete Newton's algorithm begins by choosing $\delta_1 \geq \delta^*$, and then finding the set S that minimizes the function $f(S) - \delta a(S)$. It then sets $\delta_2 \leq \delta_1$ to the value at

which, for this set S, $f(S) - \delta a(S) = 0$. It keeps iterating while there exists a set S such that $f(S) - \delta_i a(S)$ is negative, and terminates when $f(S) - \delta^i a(S)$ on all the sets is nonnegative, returning $\delta^* = \delta_i$. We give pseudocode for the discrete Newton's algorithm in Algorithm 1.

Algorithm 1 Discrete Newton's Algorithm

$i \leftarrow 0$
$\delta_1 \leftarrow \min_{i \in V, a(\{i\})>0} {}^{f(\{i\})}/_{a(\{i\})}$
$h_0 \leftarrow 1$
while $h_i \neq 0$ **do**
  $i \leftarrow i + 1$
  $h_i \leftarrow \min_{S \subseteq V} f(S) - \delta_i a(S)$
  $S_i \leftarrow_{S \subseteq V} f(S) - \delta_i a(S)$
  $\delta_{i+1} \leftarrow {}^{f(S_i)}/_{a(S_i)}$
end while
return $\delta^* \leftarrow \delta_i$

For simplicity throughout the rest of this paper, we define $k_i(S) = f(S) - \delta_i a(S)$.

We visualize the discrete Newton's algorithm by considering the graph of each function $f(S_i) - \delta a(S_i)$ in the parameter δ shown in Figure 1.

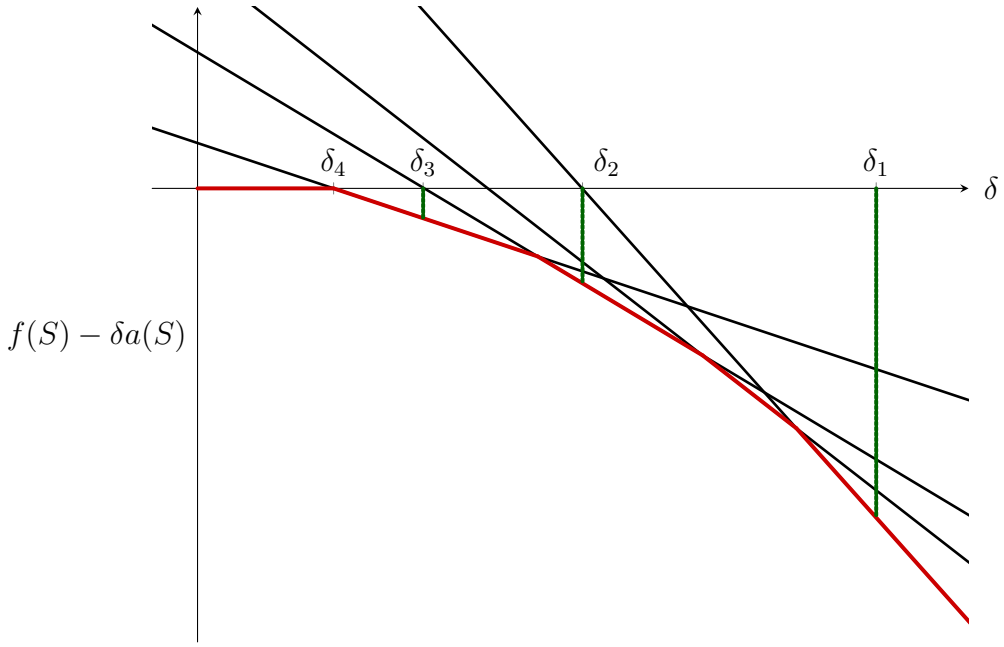

Figure 1: The visualization of the discrete Newton's algorithm.

The lower envelope of these lines are coloured in red, and the points along this line where the slope changes are called *breakpoints*. Note that this curve is piecewise linear, continuous, and concave. We note that the number of breakpoints is always greater than or equal to the number of iterations, as in every iteration we must select a new line on the envelope. This corresponds to selecting a distinct set every time, and based on the operation of the discrete Newton's algorithm, it can only select any set once.

A lemma that is easy to observe is the following.

Lemma 3 ([3])**.** *Discrete Newton's Algorithm terminates in a finite number of steps t and generates sequences:*

1. $h_1 < h_2 < \cdots < h_{t-1} < h_t = 0$,
2. $\delta_1 > \delta_2 > \cdots > \delta_{t-1} > \delta_t = \delta^* \geq 0$,
3. $a_1(S_1) > a_2(S_2) > \cdots > a_{t-1}(S_{t-1}) > a_t(S_t) \geq 0$.

Furthermore, if $a_t(S_t) > 0$ *then* $\delta^* = 0$.

The proof of Lemma 3 can be found in [3], and is omitted here.

We divide our analysis by considering when the vector a is nonnegative and when it is arbitrary, as there are different upper bounds known in these two instances. When a is nonnegative, we know that the discrete Newton's algorithm takes at most $n + 1$ iterations. When a is an arbitrary vector, on the other hand, the discrete Newton's algorithm takes at most $n^2 + O(n \log^2(n))$ iterations.

1.3 Outline

In section 2, we consider the case when we have a nonnegative vector a. We present the proof of the upper bound on the number of iterations of the discrete Newton's algorithm, a matching lower bound construction for a slightly relaxed version of the discrete Newton's algorithm and further that can be done.

In section 3, we do the same thing with an arbitrary a – we sketch the proof of the upper bound on the number of iterations of the discrete Newton's algorithm, and summarize our progress and ideas for a possible lower bound construction, as well as any observations we made.

We finish up with section 4, where we discuss future directions we could go with this problem.

1.4 Acknowledgements

We thank Michel Goemans for suggesting this project, and Yuchong Pan for mentoring us. We also thank the organizers of the Summer Program in Undergraduate Research, David Jerison and Ankur Moitra, for making this program happen and meeting with us weekly to advise us on our project.

2 Nonnegative Vectors a

It is known ([9],[4]) that if a is nonnegative, then the discrete Newton's algorithm terminates in at most $n + 1$ iterations. We reiterate the proof of this upper bound for completeness, and then present our construction of a tight lower bound with some relaxation of the initial choice of δ.

2.1 A Linear Upper Bound

We recreate the proof of the upper bound construction. We begin with the definition of strong quotients given in [4].

Definition 4. *Given two submodular functions f and f' with the same ground set V, we say f' is a* strong quotient *of f if $Y \subseteq Z$ implies*

$$f(Z) - f(Y) \geq f'(Z) - f'(Y).$$

We denote this as $f \to f'$.

We find that minimizers of submodular functions related by strong quotients are somewhat closed by unions and intersections.

Theorem 5 ([9])**.** *Let A and B be minimizers of f and f', respectively. If $f \to f'$, then $A \cap B$ and $A \cup B$ are minimizers of f and f', respectively.*

Proof. We first note that since B is a minimizer of f', we have $0 \leq f'(A \cup B) - f'(B)$, and that since A is a minimizer of f, we have $f(A) - f(A \cap B) \leq 0$. We also have that as $f \to f'$, $f'(A \cup B) - f'(B) \leq f(A \cup B) - f(B)$, and using the submodularity of f, $f(A \cup B) - f(B) \leq f(A) - f(A \cap B)$. Putting this all together we get that

$$0 \leq f'(A \cup B) - f'(B) \leq f(A) - f(A \cap B) \leq 0.$$

So we can conclude that $f'(A \cup B) + f(A \cap B) = f'(B) + f(A)$. As A and B are minimizers of f and f', respectively, we must have that $A \cap B$ and $A \cup B$ are minimizers of f and f', respectively. □

Corollary 6. *Let $f \to f'$, and A and B be the maximal minimizing subsets of f and f' respectively. Then, $B \subseteq A$.*

Proof. Assume, for sake of contradiction, that $B \not\subseteq A$. Then consider the set $A \cup B$. From Theorem 5, we know that this is also a minimizer. As $B \not\subseteq A$, we know $|A \cup B| > |A|$. □

We can then consider the k_i's generated by the discrete Newton's algorithm. We notice that they form a chain of strong quotients.

Lemma 7. *Let k_i come from the discrete Newton's algorithm, as defined above. Then,*

$$k_1 \leftarrow k_2 \leftarrow \cdots \leftarrow k_{l-1} \leftarrow k_l.$$

Proof. Consider k_i and k_{i+1}. Based on Lemma 3, we have that given two sets Z and Y such that $Y \subseteq Z$, $\delta_i a(Z \backslash Y) \geq d_{i+1} a(Z \backslash Y)$. This is equivalent to $\delta_i(a(Y) - a(Z)) \leq \delta_{i+1}(a(Y) - a(Z))$ because a is linear. Therefore we get,

$$(f(Z) - \delta_i a(Z)) - (f(Y) - \delta_i a(Y)) \leq (f(Z) - \delta_{i+1} a(Z)) - (f(Y) - \delta_{i+1} a(Y)).$$

This is equivalent to $k_i(Z) - k_i(Y) \leq k_{i+1}(Z) - k_{i+1}(Y)$, which then implies that $k_i \leftarrow k_{i+1}$. □

Theorem 8. *For a nonnegative, the discrete Newton's algorithm terminates in at most $n+1$ iterations.*

Proof. By Corollary 6, if we take S_i is the maximal minimizer of k_i, we get that

$$S_1 \supseteq S_2 \supseteq \cdots \supseteq S_{l-1} \supseteq S_l.$$

Additionally we note that Lemma 3 implies that the minimizing sets between two iterations of the discrete Newton's algorithm cannot be the same (as you make sure that the minimizing set from the previous iteration is now always nonnegative). The only exception is the last iteration on which it terminates (because then all the sets give nonnegative values for that δ). Thus, we actually have that

$$S_1 \supsetneq S_2 \supsetneq \cdots \supsetneq S_{l-1} \supseteq S_l,$$

and we can conclude that $l \leq n+1$, so discrete Newton's algorithm take at most $n+1$ iterations.

□

2.2 A Matching Lower Bound With a Relaxed Initial δ

We construct a modular function f and an $a \in \mathbb{R}^n$ such that the discrete Newton's algorithm takes $n+1$ iterations. However, we must take the following remark about the choice of initial δ_1 into account.

With the original discrete Newton's algorithm given in Algorithm 1 where $\delta_1 := \min_{i \in V} f\{i\}/a_i$, any construction with f modular will not give us a tight lower bound. Suppose f is defined by a vector $b \in \mathbb{R}^n$ such that $f(S) = \sum_{j \in S} b_j$. With this "good" initialization of δ_1, we have that $b_i/a_i \geq \delta_1$, so $b_i - \delta_1 a_i \geq 0$ for all $i \in V$. Hence $h_1 = \min_{S \subseteq V} f(S) - \delta_1 a(S) = \min_{S \subseteq V} \sum_{i \in S} b_i - \delta_1 a_i \geq 0$. By Lemma 1, Newton's algorithm terminates after one iteration.

Thus in order to obtain a construction taking $\Omega(n)$ iterations with the original "good" initial δ_1, we must consider a submodular f that is not modular. Empirically we have found this to be quite tricky, and hence an area for further exploration. Since the value of δ_1 is not used in the proof of Theorem 8, we relax this requirement and allow δ_1 to be sufficiently large in our construction.

Theorem 9. *If δ_1 is initialized to be sufficiently large, then there exists a nonnegative submodular function $f : 2^V \to \mathbb{R}$ on a finite ground set $V = [n]$ and a nonnegative vector $a \in \mathbb{R}^V$ for which the discrete Newton's algorithm requires exactly $n+1$ iterations.*

We first sketch the intuition behind our approach. Let the ground set be $V = [n]$. Based on the proof of Theorem 8, we want a chain of maximal minimizing sets $A_1 \supsetneq A_2 \supsetneq \cdots \supsetneq A_n$. Let $A_i = \{i, \ldots, n\}$. Our goal is to construct f and a such that A_i is the unique minimizing set on the i^{th} iteration.

Let $l_i(\delta) := f(A_i) - \delta a(A_i)$. Recall that $\delta_{i+1} = f(S_i)/a(S_i)$, so we want $l_i(\delta_{i+1}) = 0$. Assume $\delta_i = 2^{-(i-1)}$ and $l_i(\delta_1) = -2^{-(i-1)}$, so the graph of our desired l_i's is as shown in Figure 2.

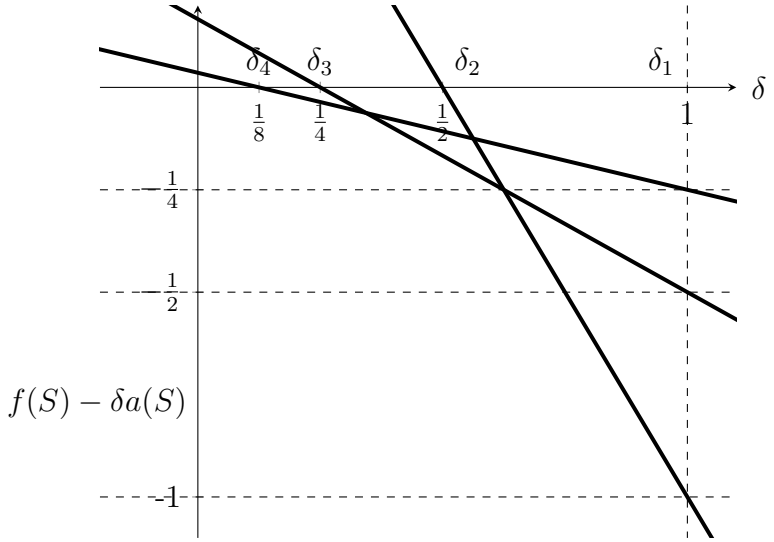

Figure 2: The graph of the desired lines l_i corresponding to the sets A_i.

To achieve such lines l_i, we want

$$f(A_i) = \frac{2}{2^i(2^i-1)}, \tag{2}$$

$$a(A_i) = \frac{2}{2^i-1}. \tag{3}$$

Proof. Our construction is as follows. Let $A_i = \{i, \ldots, n\}$. Let $f : 2^V \to \mathbb{R}$ be a modular function defined by a vector $b \in \mathbb{R}^n$ such that $f(S) = \sum_{j \in S} b_j$, where

$$b_j = \begin{cases} \frac{2^{j+2}-2^j-1}{2^j(2^j-1)(2^{j+1}-1)} & \text{if } j \in [n-1], \\ \frac{2}{2^n(2^n-1)} & \text{if } j = n. \end{cases} \tag{4}$$

Define $a \in \mathbb{R}^n$ such that

$$a_j = \begin{cases} \frac{2^{j+2}-2^{j+1}}{(2^j-1)(2^{j+1}-1)} & \text{if } j \in [n-1], \\ \frac{2}{2^n-1} & \text{if } j = n. \end{cases} \tag{5}$$

We can easily check that $b_j = f(A_j) - f(A_{j+1})$ and $a_j = a(A_j) - a(A_{j+1})$ for $j \in [n-1]$, and $b_n = f(A_n)$, $a_n = a(A_n)$. Hence (2) and (3) hold for this choice of f and a.

By straightforward calculations using (4) and (5) to find the δ-intercepts of lines, we observe that

(a) If $j \in [n-1]$, then $b_j - \delta a_j > 0$ if and only if $\delta < (2^{j+2} - 2^j - 1)/2^{2j+1}$, where $2^{-j} < (2^{j+2} - 2^j - 1)/2^{2j+1} < 2^{-(j-1)}$.

(b) If $j = n$, then $b_j - \delta a_j > 0$ if and only if $\delta < 2^{-n}$.

Initialize $\delta_1 = 1$. We show by induction that $\delta_i = 2^{-(i-1)}$ for each $i \in [n]$. The base case $i = 1$ holds by definition. Assume that $\delta_i = 2^{-(i-1)}$ for some $i \in [n]$. By our above

observation, $b_j - \delta_i a_j < 0$ for $j \geq i$ and $b_j - \delta_i a_j > 0$ for $j < i$. Hence for all $S \subseteq V$, $S \neq A_i$,

$$f(S) - \delta_i a(S) = \sum_{j \in S} (b_j - \delta_i a_j) > \sum_{j \in A_i} (b_j - \delta_i a_j) = f(A_i) - \delta_i a(A_i).$$

Then A_i is the *unique* minimizer for k_i on the ith iteration of Newton's algorithm. We also have that $h_i = f(A_i) - \delta_i a(A_i) < 0$, so $\delta_{i+1} = \frac{f(A_i)}{a(A_i)} = 2^{-i}$, completing the induction step.

If $i = n + 1$, then $\delta_{n+1} = 2^{-n}$. By our above observation again, $b_j - \delta_{n+1} a_j > 0$ for $j \in [n-1]$, and $f_n - \delta_{n+1} a_n = 0$. Hence for all $S \subseteq V$,

$$f(S) - \delta_{n+1} a(S) \geq f(A_n) - \delta_{n+1} a(A_n) = 0.$$

Thus $h_{n+1} = 0$, so Newton's algorithm terminates require exactly $n + 1$ iterations. □

2.3 Future Work

As discussed previously, the discrete Newton's algorithm terminates after exactly one iteration for modular functions f. Therefore future work in this direction includes finding a lower bound construction on which the discrete Newton's algorithm takes $\Omega(n)$ iterations despite the original "good" initialization of δ_1. One potential direction could be investigating submodular extensions of our modular construction for f, such as $g(S) = \max(f(S), k)$ for some constant k.

3 Arbitrary Vectors a

3.1 A Quadratic Upper Bound

We sketch the proof provided by Goemans et al. in [3] for the upper bound on the number of iterations the discrete Newton's algorithm takes for an arbitrary vector a. By Lemma 3, we can obtain another lemma:

Lemma 10 ([3])**.** *We define $g_i = a(S_i)$. For a discrete Newton's algorithm terminating in t iterations, for any $i < t$, we have*

$$\frac{h_{i+1}}{h_i} + \frac{g_{i+1}}{g_i} \leq 1.$$

Using Lemma 10, we partition the iterations into two types, J_g and J_h, where

$$J_g = \left\{ i \mid \frac{g_{i+1}}{g_i} \leq \frac{2}{3} \right\}, \qquad J_h = \{ i \notin J_g \}. \tag{6}$$

By considering the bound on the size of J_g, and the bound on the size of a contiguous interval of J_h, we can find a bound on the number of iterations.

A bound on J_g is given by Goemans [3] and communicated by Radzik [6].

Lemma 11 ([3]). *We have that* $|J_g| = O(n \log n)$.

Proof Sketch. If we consider $J_g = \{i_1, i_2, \ldots, i_k\}$, then we notice that the monotonicity of g, as shown in Lemma 3 implies that g is always decreasing. On top of this, since these are all elements of J_g we get that $a(S_{i_{j+1}}) \leq \frac{2}{3} a(S_{i_j})$. From this, by considering extreme points of the polytope, it can be shown that the number of sets are $k = O(n \log n)$. □

Similarly, Geomans et al. [3] shows how to bound an interval in J_h.

Lemma 12 ([3]). *Let* $[u, v] \subseteq J_h$. *Then* $[u, v] \leq n^2 + n + 1$.

Proof Sketch. We consider the submodular function k_v and show that this is a decreasing geometric sequence, through the equations in Lemma 3 and the equations for a submodular function. It then can be showed that the ring family generated by $S_{i+1}, \ldots, S_{v-1}$ does not contain S_i. We show that the maximum element in each S_i is decreasing, and hence the sets are disjoint. Then we can show that with a ground set of size $n = |V|$, the longest chain of ring families we could get is $n^2 + n + 1$. □

This gives an upper bound of $O(n \log n)O(n^2) = O(n^3 \log n)$ on the number of iterations of the discrete Newton's algorithm.

We can make this bound tighter by considering the size of J_h.

Theorem 13 ([3]). *We have*

$$|J_h| \leq n^2 + O(n \log^2 n).$$

Proof Sketch. We use a similar argument to the one used in the proof of Lemma 12. However, we use a slightly modified bound on $f(S)$ as before, and, using this, we can notice that within any interval $[u, v]$, the last $O(\log n)$ sets may already be in the family, thus reducing the estimate of $|J_h|$ to get that $|J_h| \leq n^2 + O(n \log^2 n)$. □

From this we are able to get a bound on the discrete Newton's algorithm for an arbitrary vector a.

Theorem 14 ([3]). *For a submodular function* $f : 2^n \to \mathbb{R}_+$ *and an arbitrary vector* a, *the discrete Newton's algorithm takes at most* $n^2 + O(n \log^2(n))$ *iterations.*

3.2 Lower Bounds

Based on our proof of Theorem 9, we observe the following:

Lemma 15. *The discrete Newton's algorithm will terminate within at most* $n + 1$ *iterations for any modular function* f.

Proof. Let $f : 2^V \to \mathbb{R}$ be a modular function defined by a vector $b \in \mathbb{R}^n$ such that $f(S) = \sum_{j \in S} b_j$. Since f is nonnegative and the δ's are strictly decreasing and nonnegative by Lemma 3, if $b_j - \delta_i a_j > 0$ on some iteration i, and $a_j > 0$, then $b_j - \delta_k a_j > 0$ for all subsequent iterations $k \geq i$.

Observe that since $h_i = \min_{S \subseteq V} f(S) - \delta_i a(S) = \min_{S \subseteq V} (\sum_{i \in S} f_i - \delta_i a_i)$, then S_i must consist only of $j \in [n]$ such that $b_j - \delta_i a_j \leq 0$. This also implies that $a_j > 0$.

Consider $x := \max_{j \in S_i} b_j/a_j$, and suppose $y := \arg\max_{j \in S_i} b_j/a_j$. Notice that $b_y - \delta a_y \geq 0$ for all $\delta \leq x$. We have that $b_j - x a_j \leq 0$ for all $j \in S_i$. Then $f(S_i) - x a(S_i) \leq 0$, so $\delta_{i+1} = f(S_i)/a(S_i) \leq x$ and $b_y - \delta_{i+1} a_y \geq 0$. Hence after each iteration i, we must have $b_j - \delta_{i+1} a_j \geq 0$ for at least one additional $j \in [n]$. Then $b_j - \delta_{i+1} a_j \geq 0$ for all $j \in [n]$ after at most n iterations, at which point the discrete Newton's algorithm terminates. □

Consequently, if we want a super-linear lower bound on the number of iterations in the general case, we must turn our attention to submodular functions f that are not modular.

3.2.1 Interval Submodular Functions

We conjecture that *interval submodular functions* introduced in [3] are promising candidates for lower bound constructions, as they were used in the proof of Theorem 14 to show a quadratic tight lower bound on the length of a geometrically increasing sequence of sets $S_1, \ldots, S_k$ such that $f(S_i) \geq 4 f(S_{i-1})$.

We define interval submodular functions as follows. For each $i, j \in [n]$ with $i \leq j$, we define an *interval* $[i, j] := \{k \in [n] \mid i \leq k \leq j\}$. Let the set of all intervals be $\mathcal{I} := \{[i, j] \mid i, j \in [n], i \leq j\}$. A set function $f : \mathcal{I} \to \mathbb{R}_+$, $f(\emptyset) = 0$ is *submodular on intervals* if for any $A, B \in \mathcal{I}$ such that $A \cup B \in \mathcal{I}$, $A \cap B \in \mathcal{I}$,

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B).$$

For any $S \subseteq [n]$, define $\mathcal{I}(S)$ to be the set of maximal intervals contained in S. For example, if $S = \{1, 2, 3, 6, 9, 10\}$, then $\mathcal{I}(S) = \{[1, 3], [6, 6], [9, 10]\}$. Using the following lemma, we can extend a function that is submodular on intervals to a (fully) submodular function.

Lemma 16 ([3])**.** *If $f : \mathcal{I} \to \mathbb{R}_+$ such that $f(\emptyset) = 0$ is submodular on intervals, then the function $g : 2^{[n]} \to \mathbb{R}_+$ defined by*

$$g(S) = \sum_{I \in \mathcal{I}(S)} f(I),$$

is submodular over the ground set $[n]$.

Moreover, the following lemma helps construct a function that is submodular on intervals.

Lemma 17 ([3])**.** *If $\tau, \kappa : [n] \to \mathbb{R}_+$ are monotonically increasing, then the function $f : \mathcal{I} \to \mathbb{R}_+$ defined by*

$$f([i, j]) = \tau(i)\kappa(j),$$

is submodular on intervals.

By the above two lemmas, [3] considers $\tau(i) = 4^i$ and $\kappa(j) = 4^{j(j-1)/2}$ and shows that the submodular function g induced by τ and κ gives a sequence of $\binom{n+1}{2} + 1$ sets A_i such that $g(A_i) \geq 4 g(A_{i-1})$.

3.2.2 Necessary Conditions

Although it is generally hard to derive conditions that are both necessary and sufficient, it is sometimes helpful to consider necessary conditions that must be satisfied in order to achieve $\omega(n)$ iterations. In the proof of Theorem 8, the strict nesting of maximal minimizers heavily constrains the number of iterations of the discrete Newton's algorithm. We find a necessary condition that must hold in order for the strong quotient argument to fail and potentially allow us to construct a longer chain of minimizers by avoiding strict nesting.

Consider minimizers S_i and S_j, for $i < j$. By the submodularity of f, the modularity of a, and regrouping terms, we have that

$$\begin{aligned}
&f(S_i) - \delta_i a(S_i) + f(S_j) - \delta_j a(S_j) \\
&\geq f(S_i \cup S_j) + f(S_i \cap S_j) - \delta_i a(S_i) - \delta_j a(S_j) \\
&= f(S_i \cup S_j) + f(S_i \cap S_j) - \delta_i (a(S_i \cup S_j) - a(S_j \setminus S_i)) - \delta_j (a(S_i \cup S_j) + a(S_j \setminus S_i)) \\
&= (f(S_i \cup S_j) - \delta_i a(S_i \cup S_j)) + (f(S_i \cap S_j) - \delta_j a(S_i \cap S_j)) + (\delta_i - \delta_j) a(S_j \setminus S_i). \qquad (7)
\end{aligned}$$

By Lemma 3, $\delta_i > \delta_j$, so $\delta_i - \delta_j > 0$ in the last term of (7). Recall that S_i minimizes k_i and S_j minimizes k_j. The case where $a(S_j \setminus S_i) \geq 0$ implies that $S_i \cup S_j$ minimizes k_i and $S_i \cap S_j$ minimizes k_j. This implication is used in the proof of Theorem 8 to derive the containment of maximal minimizers. Hence to avoid this, we would like the condition $a(S_j \setminus S_i) < 0$ to hold at least some of the time. By a similar derivation, we would also like $a(S_i \setminus S_j) > 0$ to hold some of the time.

In order to obtain $\omega(n)$ iterations in the case of arbitrary a, it is also necessary for some of the minimizers S_i to contain negative elements; i.e. elements j such that $a_j < 0$. Otherwise we reduce to the case of nonnegative a, which is upper bounded by $n + 1$ iterations. What conditions need to be satisfied in order for the discrete Newton's algorithm to choose a set containing negative elements as opposed to a set that does not? Since including elements j where $a_j < 0$ decreases the value of a on the set, we would need the value of f to decrease sufficiently as well. Suppose $A \subseteq V$ consists only of j such that $a_j \geq 0$, and suppose $B \subseteq V$ consists only of j such that $a_j < 0$. Then in order for B to potentially be included in the minimizer S_i, it would be necessary for $f(A \cup B) - \delta_i a(A \cup B) < f(A) - \delta_i a(A)$. Rearranging gives $f(A) - f(A \cup B) > \delta(a(A) - a(A \cup B))$. Since $a(S_i) \geq 0$, this implies that $f(A \cup B) < f(A)$.

For the submodular extension of $f([i, j]) = 4^i 4^{j(j-1)/2}$ in particular, we observe that the main ways of decreasing its value while including an additional element are by filling in gaps of size one in order to bridge disjoint but neighboring intervals (for example, $[1, 2] \cup [4]$ becomes $[1, 2, 3, 4]$) or decreasing the left endpoint of an interval (for example, $[1] \cup [4, 5]$ becomes $[1] \cup [3, 4, 5]$).

3.2.3 Dependency of a on $|V|$

Lemma 18. *If a is independent of $n = |V|$ then the discrete Newton's algorithm will take at most $O(n)$ steps.*

Proof. From Lemma 3, we know that the values of $a(S_i)$ are decreasing. Additionally, if every value of a is independent of n, then $a(S_1) = O(n)$ as it is at most n constants being

summed. Additionally, since a contains finitely many numbers, the smallest step size – the minimal difference between two values $a(S_i)$ and $a(S_{i+1})$ – is a given number independent of n and thus since we know that $a(S_i)$ is decreasing with a discrete step size, we know that the number of iterations it can take is at most the number of iterations we can take while keeping $a_t(S_t) \geq 0$. This is $O(n)$ steps. □

From this we can note that a constant vector a as proposed in the previous section cannot give a quadratic bound. However, we can introduce some dependency on n while keeping the structure of a the same.

3.2.4 Ideas for Minimizers

Based on computational experiments with the submodular extension of $f([i,j]) = 4^i 4^{j(j-1)/2}$, we present a guess for $a \in \mathbb{R}^n$ and the minimizing sets S_i that could potentially give $\Omega(n^2)$ iterations of the discrete Newton's algorithm. Consider $a \in \mathbb{R}^n$ such that $a_j < 0$ for $j \equiv 1 \pmod{k}$ for some integer k. We present an example for $k = 3$, $n = 9$:

$$
\begin{aligned}
S_1 &= \{2,3,5,6,8,9\} \\
S_2 &= \{2,3,5,6,7,8,9\} \\
S_3 &= \{2,3,4,5,6,7,8,9\} \\
S_4 &= \{1,2,3,4,5,6,7,8,9\} \\
S_5 &= \{2,3,5,6,8\} \\
S_6 &= \{2,3,5,6,7,8\} \\
S_7 &= \{2,3,4,5,6,7,8\} \\
&\vdots
\end{aligned}
$$

The general pattern is as follows. Assuming we initialize δ_1 to be sufficiently large (say, ∞), it is optimal for S_1 to exclude all negative elements. The next set S_2 fills in the rightmost "gap" where a negative element was excluded. Then S_3 fills in the next rightmost "gap", and so on until all the gaps are filled, as in S_4. The next set S_5 then excludes all the negative elements that were filled in while simultaneously shrinking the rightmost endpoint from 9 to 8. We then repeat the same pattern of filling in the gaps one-by-one starting from the right, then removing the gaps and shrinking the right endpoint by one. We stop when we reach a sufficiently small set, such as $\{1,2,3\}$. Observe that the length of this sequence of sets is $\Theta(n^2)$, since if the rightmost endpoint is j, then it takes $\lceil j/k \rceil$ sets to fill in the gaps before the endpoint is shrunk to $j-1$. Then the total length of the sequence is $k \cdot \lceil n/k \rceil + k \cdot (\lceil n/k \rceil - 1) + \cdots + 2k + 1 = \Theta(n^2)$.

We observe this pattern only generally for a specific example we tested, in which $a = (-1,5,5,-1,5,5,-1\ldots)$. For this particular case, oftentimes steps in this pattern are skipped; for example, not all the gaps may be filled in before the right endpoint is decreased, or several gaps may be filled simultaneously. If we can find a vector a such that the pattern is followed more closely, perhaps with slight adjustments such as filling a constant number of gaps on each iteration, or shrinking the right endpoint by a constant number of elements, then we could obtain $\Omega(n^2)$ iterations.

We give some intuition for why this pattern might be reasonable. As mentioned in Section 3.2.4, in order to include a negative element in the minimizer, we need the value of f to decrease. Filling in the gaps is one way of doing so. We also observe that this satisfies the condition $a(S_j \setminus S_i) < 0$ for $i < j$. Removing all the gaps increases the value of f despite the value of δ having decreased during the round, so decreasing the right endpoint compensates in a way. Furthermore, regularly spacing the negative elements guarantees an even distribution of them despite shrinking the right endpoint.

3.2.5 Future Work on Lower Bound Constructions for Arbitrary a

Finally, we present an outline of what a potential lower bound construction and proof may entail. As in the proof of Theorem 9, a geometric picture of the lines that contribute to the piecewise linear function $h(\delta) = \min_S f(S) - \delta a(S)$ could help guide us toward an appropriate f and a, as their values on each minimizer S_i will be defined by the y-intercepts and slopes, respectively, of the lines corresponding to the minimizers S_i. This geometric picture can also help us define values for the δ_i's, which will be a subset of the δ-intercepts of the lines. A particularly nice picture would be one in which all lines correspond to the minimizers S_i, as was the case in the proof of Theorem 9. We would also need to construct sets A_i that are candidates for minimizers that agree with the values derived from the picture.

Ideally we would want to work with interval submodular functions, as they only require $O(n)$ input parameters, as opposed to an exponential number of input parameters to define a general submodular function. Once we have defined f, a, and each δ_i and A_i, it is not immediately obvious for the case of f not modular how to prove that A_i is indeed the minimizer of k_i among an exponential number of other subsets of V. However, we can use the following duality theorem from [2].

Theorem 19 ([2])**.** *Let $f : 2^V \to \mathbb{R}$ be a submodular function on a finite ground set $V = [n]$. Then*

$$\min_S f(S) = \max_{x \in P(f)} \sum_{j \in V} \min(x_j, 0). \tag{8}$$

We give the proof of one direction of this equality, which is relevant for our purposes. For any $S \subseteq V$ and any $x \in P(f)$, we have that

$$\sum_{j \in V} \min(x_j, 0) \leq \sum_{j \in S} \min(x_j, 0) \leq \sum_{j \in V} x_j \leq f(S),$$

since $\min(x_j, 0) \leq 0$ and $x \in P(f)$.

To prove that A_i minimizes k_i, we simply need to present an $x \in P(f)$ such that we achieve equality in (8); i.e. $f(A_i) = \sum_{j \in V} \min(x_j, 0)$. To show that a given $x \in \mathbb{R}^n$ belongs to $P(f)$, we can show that it is a convex combination of vertices of $P(f)$ via the following two theorems from [5].

Theorem 20 ([5])**.** *Let π be a permutation of $[n]$. Define $x^{(\pi)} \in \mathbb{R}^n$ as follows:*

$$x^{(\pi)}_{\pi(1)} = f(\{\pi(1)\}),$$
$$x^{(\pi)}_{\pi(j)} = f(\{\pi(1), \ldots, \pi(j)\}) - f(\{\pi(1), \ldots, \pi(j-1)\}) \text{ for } j = 2, \ldots, n.$$

Then $x^{(\pi)} \in P(f)$.

Theorem 21 ([5]). *Given $x^* \in \mathbb{R}^n$, if there exists a set $\mathcal{S}$ of permutations such that*

$$x^* = \sum_{\pi \in \mathcal{S}} \lambda_\pi x^{(\pi)},$$

where $\sum_{\pi \in \mathcal{S}} \lambda_\pi = 1$ and $\lambda_\pi \geq 0$ for all $\pi \in \mathcal{S}$, then $x^* \in P(f)$.

The following is an example of how we might apply these theorems to our work. Consider the submodular extension of $f([i,j]) = 4^i 4^{j(j-1)/2}$ on the ground set $V = \{1,2,3\}$, and $a = (-1,5,5)$.

$S =$	$f(S)$	$l_S(\delta) = f(S) - \delta a(S)$	$l_S(\delta)$ for $\delta = 4^2$
$\emptyset$	0	0	0
$\{1\}$	4^1	$4^1 + \delta$	20
$\{1,2\}$	4^2	$4^2 - 4\delta$	-48
$\{2\}$	4^3	$4^3 - 5\delta$	-16
$\{1,2,3\}$	4^4	$4^4 - 9\delta$	112
$\{2,3\}$	4^5	$4^5 - 10\delta$	864
$\{3\}$	4^6	$4^6 - 5\delta$	4016

The set $\{1,2\}$ is the unique minimizer of $f(S) - \delta a(S)$ for $\delta = 4^2$. Assuming we only had knowledge of $f(\{1,2\}) - 4^2 a(\{1,2\}) = -48$, we can prove that this is the minimum as follows. Consider the permutation $\pi = (2,1,3)$. By Theorem 20, we define x^{213} as follows:

$$\begin{aligned} x_2^{213} &= -16 < 0, \\ x_1^{213} &= -48 - (-16) < 0, \\ x_3^{213} &= 112 - (-48) > 0. \end{aligned}$$

Hence, $\sum_{j=1}^{3} \min(x_j^{213}, 0) = -48$, which proves that $\{1,2\}$ is the minimizer. We also note that if a submodular function has multiple minimizers, then we have to consider multiple permutations on $[n]$.

4 Future Directions

In addition to future work on lower bound constructions as mentioned in previous sections, there are many other problems related to the discrete Newton's algorithm that remain open.

Firstly, the number of breakpoints on the lower envelope $h(\delta) = \min_S f(S) - \delta a(S)$ can give us another insight into the the number of iterations for the discrete Newton's algorithm. In any instance of the algorithm, the number of iterations is upper bounded by the number of breakpoints plus one, because each iteration of the algorithm passes at least one breakpoint (this can be easily seen by Figure 1). By looking at the breakpoints, one might be able to find better upper bounds for the discrete Newton's algorithm. To the best of our knowledge, nothing is known about the number of breakpoints in the literature. Hence, any linear, quadratic, polynomial or exponential bound would be interesting.

We can also consider the case where a is not necessarily modular. In particular, we might be interested in the case where a is the difference of two cut functions in an undirected graph

(which are submodular), because this corresponds to the parametric maximum flow problem in which the edge capacities are linear in the parameter.

There are also variants on the discrete Newton's algorithm, such as the *accelerated* discrete Newton's algorithm by Dadush et al. [1] and another variant which solves the quickest transshipment problem [8]. These variants are also proved to have quadratic upper bounds, but we have no knowledge whether there is a matching lower bound.

References

[1] Daniel Dadush, Zhuan Khye Koh, Bento Natura, and László A Végh. An accelerated newton-dinkelbach method and its application to two variables per inequality systems. *arXiv preprint arXiv:2004.08634*, 2020.

[2] J. Edmonds. Submodular functions, matroids, and certain polyhedra. In *Guy, R., Hanani, H.,Sauer, N., Schönheim, J. (eds.) Combinatorial Structures and Their Applications*. Gordon and Breach, 1970.

[3] Michel X Goemans, Swati Gupta, and Patrick Jaillet. Discrete newton's algorithm for parametric submodular function minimization. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 212–227. Springer, 2017.

[4] Satoru Iwata, Kazuo Murota, and Maiko Shigeno. A fast parametric submodular intersection algorithm for strong map sequences. *Mathematics of Operations Research*, 22(4):803–813, 1997.

[5] László Lovász. Submodular functions and convexity. In *Mathematical programming the state of the art*, pages 235–257. Springer, 1983.

[6] Tomasz Radzik. Fractional combinatorial optimization. In *Handbook of combinatorial optimization*, pages 429–478. Springer, 1998.

[7] Miriam Schlöter. *Flows over time and submodular function minimization*. Technische Universitaet Berlin (Germany), 2018.

[8] Miriam Schlöter, Martin Skutella, and Khai Van Tran. A faster algorithm for quickest transshipments via an extended discrete newton method. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 90–102. SIAM, 2022.

[9] Donald M Topkis. Minimizing a submodular function on a lattice. *Operations research*, 26(2):305–321, 1978.