Submodular Considerations for Graph Density

Sambuddha Roy

October 27, 2010

Graph Density

- This talk is going to be about graph density and its variants.
- Short definition: given a graph G, graph density $d(G)$ is defined as $\max_{H \subseteq G} \frac{|E(H)|}{|V(H)|}$.

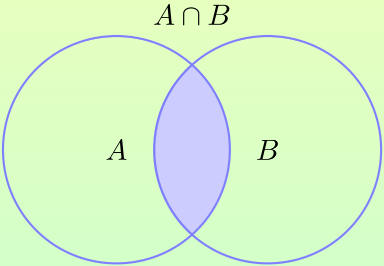
General Goals

- We want to *optimize* various functions appearing in real life scenarios.
- However, quite a few of the optimization questions from real life scenarios are *hard*.
- The good news is that we still have a lot of optimization questions that indeed can be solved fast.

Definition of Submodular Functions

- A function $f : 2^{[U]} \to \mathbb{R}$ is *submodular* [6, 13] iff for every $A, B \subseteq U$, it holds that

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



We will give a few alternate definitions of submodular functions.

- **Law of Diminishing Returns**. Alternately, a function f is submodular iff for every $A \subseteq B \subseteq U$ and an element $t \in U \backslash B$, it holds that
$$f(A+t) - f(A) \geq f(B+t) - f(B)$$
- **Law of Marginal Utilities**. A function f is submodular iff for every $A \subseteq U$ and $t, u \in U \backslash A$, it holds that
$$f(A+t) - f(A) \geq f(A+u+t) - f(A+u)$$

Definition of Submodular Functions

We will give a few alternate definitions of submodular functions.

- **Law of Diminishing Returns**. Alternately, a function f is submodular iff for every $A \subseteq B \subseteq U$ and an element $t \in U \backslash B$, it holds that

$$f(A+t) - f(A) \geq f(B+t) - f(B)$$

- **Law of Marginal Utilities**. A function f is submodular iff for every $A \subseteq U$ and $t, u \in U \backslash A$, it holds that

$$f(A+t) - f(A) \geq f(A+u+t) - f(A+u)$$

Modular, Supermodular?

- A function $f : 2^{[U]} \to \mathbb{R}$ is *modular* iff for every $A, B \subseteq U$, it holds that
$$f(A) + f(B) = f(A \cup B) + f(A \cap B)$$
- A function $f : 2^{[U]} \to \mathbb{R}$ is *supermodular* iff for every $A, B \subseteq U$, it holds that
$$f(A) + f(B) \leq f(A \cup B) + f(A \cap B)$$

Modular, Supermodular?

- A function $f : 2^{[U]} \to \mathbb{R}$ is *modular* iff for every $A, B \subseteq U$, it holds that
$$f(A) + f(B) = f(A \cup B) + f(A \cap B)$$

- A function $f : 2^{[U]} \to \mathbb{R}$ is *supermodular* iff for every $A, B \subseteq U$, it holds that
$$f(A) + f(B) \leq f(A \cup B) + f(A \cap B)$$

Modular, Supermodular

- Modular functions are in some sense trivial. For instance, given a set A, the *cardinality* function $f(A) = |A|$ is a modular function.
- In fact, *all* modular functions are (somewhat) of this type.
- Examples of Supermodular functions? Given a graph $G = (V, E)$, and any subset $S \subseteq V$ of vertices, consider the set $E(S) \subseteq E$ which consists of the edges *inside* the set S. Then we can define the function $f(S) = |E(S)|$. We can easily prove that this function f is *supermodular*.

Modular, Supermodular

- Modular functions are in some sense trivial. For instance, given a set A, the *cardinality* function $f(A) = |A|$ is a modular function.
- In fact, *all* modular functions are (somewhat) of this type.
- Examples of Supermodular functions? Given a graph $G = (V, E)$, and any subset $S \subseteq V$ of vertices, consider the set $E(S) \subseteq E$ which consists of the edges *inside* the set S. Then we can define the function $f(S) = |E(S)|$. We can easily prove that this function f is *supermodular*.

Operations on functions

- Suppose we have a submodular function f. Then $-f$ is supermodular.
- Changing (increasing/decreasing) by a constant does not change the modularity of a function.
- If f and g are two submodular functions, then the function $f + g$ defined as $(f + g)(S) = f(S) + g(S)$ is also submodular.
- Thus, if f is submodular, and g supermodular, then the function $f - g$ is submodular.

Why the interest in submodular functions

- Economics: As the “laws" indicate, these apply to economics widely – the more you have of some quantity, the lesser the benefit any extra unit gives you.
- In computer science, submodular functions are ubiquitous. Given an undirected graph G, let $\delta(S)$ denote the *cut* function – this refers to the edges between the set S and its complement $\bar{S}$. It is commonly known that the cut function in undirected graphs is submodular (similarly, in directed graphs, the *directed* cut function is submodular).
- Lots of other examples abound: generalized max flow, *coverage* in set systems, the neighborhood function $N(Z)$ for a set $Z \subseteq A$ in a bipartite graph $G = (A, B)$.

Why the interest in submodular functions

- The *principal* interest in submodular functions (apart from the fact, that they model many real life settings) is that they can actually be *minimized* in polynomial time!
- For instance, it was well known (the Ford-Fulkerson result [5]) that graph cuts can be minimized in polynomial time.
- The general result for submodular functions is comparatively much more recent – Grotschel, Lovasz, Schrijver proved [8] that submodular functions may be minimized in polynomial time.

Quick proof that $\delta(S)$ is submodular

- Given a graph G, we know that $\delta(S) = |E(G)| - |E(S)| - |E(\bar{S})|$.
- Suppose we are given a supermodular function $g(S)$, is the function $h(S) = g(S) + g(\bar{S})$ supermodular too?
- Calculation: $h(A) + h(B) = g(A) + g(B)$

  $+ g(\bar{A}) + g(\bar{B}) \leq g(A \cap B) + g(A \cup B) + g(\bar{A} \cap \bar{B}) + g(\bar{A} \cup \bar{B})$.
- But this last is $= g(A \cap B) + g(\overline{A \cap B}) + g(A \cup B) + g(\overline{A \cup B}) = h(A \cup B) + h(A \cap B)$.
- Thus, the function $|E(S)| + |E(\bar{S})|$ is supermodular. Hence $\delta(S)$ is submodular.

Quick proof that $\delta(S)$ is submodular

- Given a graph G, we know that $\delta(S) = |E(G)| - |E(S)| - |E(\bar{S})|$.
- Suppose we are given a supermodular function $g(S)$, is the function $h(S) = g(S) + g(\bar{S})$ supermodular too?
- Calculation: $h(A) + h(B) = g(A) + g(B)$

  $+\, g(\bar{A}) + g(\bar{B}) \leq g(A \cap B) + g(A \cup B) + g(\bar{A} \cap \bar{B}) + g(\bar{A} \cup \bar{B})$.
- But this last is $= g(A \cap B) + g(\overline{A \cap B}) + g(A \cup B) + g(\overline{A \cup B}) = h(A \cup B) + h(A \cap B)$.
- Thus, the function $|E(S)| + |E(\bar{S})|$ is supermodular. Hence $\delta(S)$ is submodular.

Quick proof that $\delta(S)$ is submodular

- Given a graph G, we know that $\delta(S) = |E(G)| - |E(S)| - |E(\bar{S})|$.
- Suppose we are given a supermodular function $g(S)$, is the function $h(S) = g(S) + g(\bar{S})$ supermodular too?
- Calculation: $h(A) + h(B) = g(A) + g(B)$

  $+ g(\bar{A}) + g(\bar{B}) \leq g(A \cap B) + g(A \cup B) + g(\bar{A} \cap \bar{B}) + g(\bar{A} \cup \bar{B})$.
- But this last is

  $= g(A \cap B) + g(\overline{A \cap B}) + g(A \cup B) + g(\overline{A \cup B}) = h(A \cup B) + h(A \cap B)$.
- Thus, the function $|E(S)| + |E(\bar{S})|$ is supermodular. Hence $\delta(S)$ is submodular.

Quick proof that $\delta(S)$ is submodular

- Given a graph G, we know that $\delta(S) = |E(G)| - |E(S)| - |E(\bar{S})|$.
- Suppose we are given a supermodular function $g(S)$, is the function $h(S) = g(S) + g(\bar{S})$ supermodular too?
- Calculation: $h(A) + h(B) = g(A) + g(B)$

  $+ g(\bar{A}) + g(\bar{B}) \leq g(A \cap B) + g(A \cup B) + g(\bar{A} \cap \bar{B}) + g(\bar{A} \cup \bar{B})$.
- But this last is

  $= g(A \cap B) + g(\overline{A \cap B}) + g(A \cup B) + g(\overline{A \cup B}) = h(A \cup B) + h(A \cap B)$.
- Thus, the function $|E(S)| + |E(\bar{S})|$ is supermodular. Hence $\delta(S)$ is submodular.

Connections to convexity

- A function $f : X \to \mathbb{R}$ is *convex* iff $f(a) + f(b) \geq f(\lambda \cdot a + (1 - \lambda) \cdot b)$ for any $\lambda \in [0, 1]$ (and any $a, b \in X$).
- Why are convex functions of interest? Again, because they can be *minimized*!

Connections to convexity

- A function $f : X \to \mathbb{R}$ is *convex* iff $f(a) + f(b) \geq f(\lambda \cdot a + (1 - \lambda) \cdot b)$ for any $\lambda \in [0, 1]$ (and any $a, b \in X$).
- Why are convex functions of interest? Again, because they can be *minimized*!

More about convexity

- Consider a function $f : \mathbb{R}^n \to \mathbb{R}$.
- A convex function has a *unique* optimum – the local optimum is the global optimum (minimum).
- So given the curve of a convex function, we can consider *gradient descent* and thereby find the optimal point for which the function is minimized.

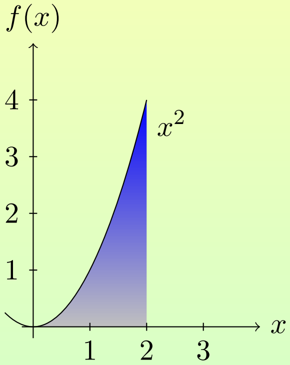
More about convexity

- Consider a function $f : \mathbb{R}^n \to \mathbb{R}$.
- A convex function has a *unique* optimum – the local optimum is the global optimum (minimum).
- So given the curve of a convex function, we can consider *gradient descent* and thereby find the optimal point for which the function is minimized.

More about convexity

- Consider a function $f : \mathbb{R}^n \to \mathbb{R}$.
- A convex function has a *unique* optimum – the local optimum is the global optimum (minimum).
- So given the curve of a convex function, we can consider *gradient descent* and thereby find the optimal point for which the function is minimized.

More about convexity



- A convex function.
- Here, $f(0) = 0$, $f(2) = 4$. Notice that $f(1) = f(\frac{0+2}{2}) = 1 \leq \frac{f(0)+f(2)}{2}$.

Bottomline

- The bottomline then is that convex functions can be *minimized* (in polynomial time?)
- What does this have to do with submodular functions?

The Lovász Extension

- Given a (set) function $f : \{0,1\}^{[U]} \to \mathbb{R}$, Lovász constructs a *smooth* function $F(x) : [0,1]^{|U|} \to \mathbb{R}$ such that the following holds:
  - The function f is submodular iff the function F is convex.
  - The values of F *coincide* with that of f on $0-1$ vectors.
  - Minimizing the function f corresponds to minimizing the function F.

The Lovász Extension

- Given a (set) function $f : \{0,1\}^{[U]} \rightarrow \mathbb{R}$, Lovász constructs a *smooth* function $F(x) : [0,1]^{|U|} \rightarrow \mathbb{R}$ such that the following holds:
  - The function f is submodular iff the function F is convex.
  - The values of F *coincide* with that of f on $0-1$ vectors.
  - Minimizing the function f corresponds to minimizing the function F.

The Lovász Extension

- Given a (set) function $f : \{0,1\}^{[U]} \to \mathbb{R}$, Lovász constructs a *smooth* function $F(x) : [0,1]^{|U|} \to \mathbb{R}$ such that the following holds:
  - The function f is submodular iff the function F is convex.
  - The values of F *coincide* with that of f on $0-1$ vectors.
  - Minimizing the function f corresponds to minimizing the function F.

The Lovász Extension

- Given a (set) function $f : \{0,1\}^{[U]} \to \mathbb{R}$, Lovász constructs a *smooth* function $F(x) : [0,1]^{|U|} \to \mathbb{R}$ such that the following holds:
  - The function f is submodular iff the function F is convex.
  - The values of F *coincide* with that of f on $0-1$ vectors.
  - Minimizing the function f corresponds to minimizing the function F.

The Lovász Extension

- So, for any submodular function f, there corresponds a convex function F.
- The correspondence is one-way: for a general convex function F, there *may not* correspond a submodular function f.
- In fact this lack of correspondence actually helps!

The Lovász Extension

- So, for any submodular function f, there corresponds a convex function F.
- The correspondence is one-way: for a general convex function F, there *may not* correspond a submodular function f.
- In fact this lack of correspondence actually helps!

The Lovász Extension

- So, for any submodular function f, there corresponds a convex function F.
- The correspondence is one-way: for a general convex function F, there *may not* correspond a submodular function f.
- In fact this lack of correspondence actually helps!

Optimization of submodular functions: Minimization

- We have seen so far that submodular functions can be *minimized* in polynomial time.
- Caveat: The polynomial time is **HUGE**! Given a submodular function on $\{0,1\}^n$, the strongly polynomial time algorithm of Schrijver [15] achieves $O(n^7)$ time!
- However, this is mostly used as a "proof-of-concept". For specific submodular functions (like the cut function $\delta(S)$), we can then search for faster algorithms.
- Open Question: Can we approximately minimize a submodular function fast?

Optimization of submodular functions: Minimization

- We have seen so far that submodular functions can be *minimized* in polynomial time.
- Caveat: The polynomial time is **HUGE**! Given a submodular function on $\{0,1\}^n$, the strongly polynomial time algorithm of Schrijver [15] achieves $O(n^7)$ time!
- However, this is mostly used as a "proof-of-concept". For specific submodular functions (like the cut function $\delta(S)$), we can then search for faster algorithms.
- Open Question: Can we approximately minimize a submodular function fast?

Optimization of submodular functions: Minimization

- We have seen so far that submodular functions can be *minimized* in polynomial time.
- Caveat: The polynomial time is **HUGE**! Given a submodular function on $\{0,1\}^n$, the strongly polynomial time algorithm of Schrijver [15] achieves $O(n^7)$ time!
- However, this is mostly used as a “proof-of-concept". For specific submodular functions (like the cut function $\delta(S)$), we can then search for faster algorithms.
- Open Question: Can we approximately minimize a submodular function fast?

Optimization of submodular functions: Minimization

- We have seen so far that submodular functions can be *minimized* in polynomial time.
- Caveat: The polynomial time is **HUGE**! Given a submodular function on $\{0,1\}^n$, the strongly polynomial time algorithm of Schrijver [15] achieves $O(n^7)$ time!
- However, this is mostly used as a “proof-of-concept". For specific submodular functions (like the cut function $\delta(S)$), we can then search for faster algorithms.
- Open Question: Can we approximately minimize a submodular function fast?

Optimization of submodular functions: Maximization

- Note that the correspondence between submodular functions and convex functions is only one-way.
- We cannot expect a general convex function to be approximated to any reasonable factor. Imagine a convex function on n variables.
- However, this does not really create a hurdle for maximization of submodular functions.
- Of course, this problem is NP-hard. Consider the question of MAXCUT for instance.

Optimization of submodular functions: Maximization

- Note that the correspondence between submodular functions and convex functions is only one-way.
- We cannot expect a general convex function to be approximated to any reasonable factor. Imagine a convex function on n variables.
- However, this does not really create a hurdle for maximization of submodular functions.
- Of course, this problem is NP-hard. Consider the question of $\mathtt{MAXCUT}$ for instance.

Optimization of submodular functions: Maximization

- Note that the correspondence between submodular functions and convex functions is only one-way.
- We cannot expect a general convex function to be approximated to any reasonable factor. Imagine a convex function on n variables.
- However, this does not really create a hurdle for maximization of submodular functions.
- Of course, this problem is NP-hard. Consider the question of MAXCUT for instance.

Optimization of submodular functions: Maximization

- Note that the correspondence between submodular functions and convex functions is only one-way.
- We cannot expect a general convex function to be approximated to any reasonable factor. Imagine a convex function on n variables.
- However, this does not really create a hurdle for maximization of submodular functions.
- Of course, this problem is NP-hard. Consider the question of $\texttt{MAXCUT}$ for instance.

Optimization of submodular functions: Maximization

- A general (i.e. non-monotone) submodular function may be *easily* approximated within constant factors.
- A randomized algorithm: pick a random set; this achieves a factor of $\frac{1}{4}$.
- Feige et al [4] have shown a $\frac{2}{5}$ approximation algorithm. Vondrák has since improved this factor to 0.41.
- Hardness: Since we are considering general submodular functions, they cannot be maximized to a factor better than (essentially) $\frac{1}{2}$.

Optimization of submodular functions: Maximization

- A general (i.e. non-monotone) submodular function may be *easily* approximated within constant factors.
- A randomized algorithm: pick a random set; this achieves a factor of $\frac{1}{4}$.
- Feige et al [4] have shown a $\frac{2}{5}$ approximation algorithm. Vondrák has since improved this factor to 0.41.
- Hardness: Since we are considering general submodular functions, they cannot be maximized to a factor better than (essentially) $\frac{1}{2}$.

Optimization of submodular functions: Maximization

- A general (i.e. non-monotone) submodular function may be *easily* approximated within constant factors.
- A randomized algorithm: pick a random set; this achieves a factor of $\frac{1}{4}$.
- Feige et al [4] have shown a $\frac{2}{5}$ approximation algorithm. Vondrák has since improved this factor to 0.41.
- Hardness: Since we are considering general submodular functions, they cannot be maximized to a factor better than (essentially) $\frac{1}{2}$.

Optimization of submodular functions: Maximization

- A general (i.e. non-monotone) submodular function may be *easily* approximated within constant factors.
- A randomized algorithm: pick a random set; this achieves a factor of $\frac{1}{4}$.
- Feige et al [4] have shown a $\frac{2}{5}$ approximation algorithm. Vondrák has since improved this factor to 0.41.
- Hardness: Since we are considering general submodular functions, they cannot be maximized to a factor better than (essentially) $\frac{1}{2}$.

Optimization of submodular functions: Maximization

- These algorithms are quite fast; often they involve a greedy subroutine, or a local search method.
- Improved factors/algorithms exist for the special case of *monotone* submodular functions. Maximizing a monotone submodular function is a no-brainer: just pick the entire set. So?
- Consider the maximization questions with other constraints called knapsack constraints or matroid constraints. Recent flurry of work shows constant factor (small constant), fast approximation algorithms for these problems too. See [11, 3, 1, 12].

Optimization of submodular functions: Maximization

- These algorithms are quite fast; often they involve a greedy subroutine, or a local search method.
- Improved factors/algorithms exist for the special case of *monotone* submodular functions. Maximizing a monotone submodular function is a no-brainer: just pick the entire set. So?
- Consider the maximization questions with other constraints called knapsack constraints or matroid constraints. Recent flurry of work shows constant factor (small constant), fast approximation algorithms for these problems too. See [11, 3, 1, 12].

Optimization of submodular functions: Maximization

- These algorithms are quite fast; often they involve a greedy subroutine, or a local search method.
- Improved factors/algorithms exist for the special case of *monotone* submodular functions. Maximizing a monotone submodular function is a no-brainer: just pick the entire set. So?
- Consider the maximization questions with other constraints called knapsack constraints or matroid constraints. Recent flurry of work shows constant factor (small constant), fast approximation algorithms for these problems too. See [11, 3, 1, 12].

A slight paradox

- While submodular functions may be *minimized* exactly, we do not know how to minimize them fast.
- On the other hand, they (perhaps) cannot be maximized in polynomial time, but we can get good approximations quite fast!

A slight paradox

- While submodular functions may be *minimized* exactly, we do not know how to minimize them fast.
- On the other hand, they (perhaps) cannot be maximized in polynomial time, but we can get good approximations quite fast!

About Graph Density

- We will define the graph density parameter.
- Motivation, applications?

About Graph Density

- We will define the graph density parameter.
- Motivation, applications?

Density of a graph

- Consider the following problem: Given a graph $G = (V, E)$, consider the density of a subset $H \subseteq V$ of vertices as $\dfrac{|E(H)|}{|H|}$.
- **Definition:** The density of a graph $d(G)$ is defined as

$$d(G) = \max_{H \subseteq V} \frac{|E(H)|}{|H|}.$$

- Goldberg [7] proved that we can compute this graph parameter in polynomial time (via max-flow computations).

Followup work on density

- Charikar [2] showed a direct LP that computes the above value. Also, Charikar shows a greedy 2-factor approximation (that runs in linear time) for the same graph parameter.
- Charikar's motivation was to prove that one may compute the *directed graph density parameter* $d'(G)$ in polynomial time, thus resolving a question raised by Kannan & Vinay [9].
- Khuller & Saha [10] give max-flow formulations and a linear time greedy algorithm for the directed graph density parameter.

An easy result

- We can state the following general result: Given a graph $G = (V, E)$, a submodular function f and a supermodular function g over subsets of V, consider the *ratio* graph parameter $f(S)/g(S)$.
- **Theorem:** We can minimize $f(S)/g(S)$ in polynomial time.

An easy result

- We can state the following general result: Given a graph $G = (V, E)$, a submodular function f and a supermodular function g over subsets of V, consider the *ratio* graph parameter $f(S)/g(S)$.
- **Theorem:** We can minimize $f(S)/g(S)$ in polynomial time.

Proof

- The proof is trivial. Consider the function $h_t(S) = f(S) - t \cdot g(S)$.
- For any specific t, the function h_t is *submodular* and can therefore be minimized in polynomial time.
- Consider the sequence of values $t = 0, 1, 2, \cdots$. Mark the point where $h_t(S)$ falls below 0.
- A more fine-tuned search gives us the value of t where this transition happens (we can speed things up by binary search).
- Thus we may compute $\min_S f(S)/g(S)$ in polynomial time.
- Alternately, given supermodular $g(S)$ and submodular $f(S)$, we may likewise compute $\max_S g(S)/f(S)$ in polynomial time.

Proof

- The proof is trivial. Consider the function $h_t(S) = f(S) - t \cdot g(S)$.
- For any specific t, the function h_t is *submodular* and can therefore be minimized in polynomial time.
- Consider the sequence of values $t = 0, 1, 2, \cdots$. Mark the point where $h_t(S)$ falls below 0.
- A more fine-tuned search gives us the value of t where this transition happens (we can speed things up by binary search).
- Thus we may compute $\min_S f(S)/g(S)$ in polynomial time.
- Alternately, given supermodular $g(S)$ and submodular $f(S)$, we may likewise compute $\max_S g(S)/f(S)$ in polynomial time.

Proof

- The proof is trivial. Consider the function $h_t(S) = f(S) - t \cdot g(S)$.
- For any specific t, the function h_t is *submodular* and can therefore be minimized in polynomial time.
- Consider the sequence of values $t = 0, 1, 2, \cdots$. Mark the point where $h_t(S)$ falls below 0.
- A more fine-tuned search gives us the value of t where this transition happens (we can speed things up by binary search).
- Thus we may compute $\min_S f(S)/g(S)$ in polynomial time.
- Alternately, given supermodular $g(S)$ and submodular $f(S)$, we may likewise compute $\max_S g(S)/f(S)$ in polynomial time.

Proof

- The proof is trivial. Consider the function $h_t(S) = f(S) - t \cdot g(S)$.
- For any specific t, the function h_t is *submodular* and can therefore be minimized in polynomial time.
- Consider the sequence of values $t = 0, 1, 2, \cdots$. Mark the point where $h_t(S)$ falls below 0.
- A more fine-tuned search gives us the value of t where this transition happens (we can speed things up by binary search).
- Thus we may compute $\min_S f(S)/g(S)$ in polynomial time.
- Alternately, given supermodular $g(S)$ and submodular $f(S)$, we may likewise compute $\max_S g(S)/f(S)$ in polynomial time.

Proof

- The proof is trivial. Consider the function $h_t(S) = f(S) - t \cdot g(S)$.
- For any specific t, the function h_t is *submodular* and can therefore be minimized in polynomial time.
- Consider the sequence of values $t = 0, 1, 2, \cdots$. Mark the point where $h_t(S)$ falls below 0.
- A more fine-tuned search gives us the value of t where this transition happens (we can speed things up by binary search).
- Thus we may compute $\min_S f(S)/g(S)$ in polynomial time.
- Alternately, given supermodular $g(S)$ and submodular $f(S)$, we may likewise compute $\max_S g(S)/f(S)$ in polynomial time.

Proof

- The proof is trivial. Consider the function $h_t(S) = f(S) - t \cdot g(S)$.
- For any specific t, the function h_t is *submodular* and can therefore be minimized in polynomial time.
- Consider the sequence of values $t = 0, 1, 2, \cdots$. Mark the point where $h_t(S)$ falls below 0.
- A more fine-tuned search gives us the value of t where this transition happens (we can speed things up by binary search).
- Thus we may compute $\min_S f(S)/g(S)$ in polynomial time.
- Alternately, given supermodular $g(S)$ and submodular $f(S)$, we may likewise compute $\max_S g(S)/f(S)$ in polynomial time.

Examples: About $E(S)$

- Given a graph $G = (V, E)$, the function $|E(S)|$ is supermodular, while $|S|$ is modular (and thus, of course, submodular). This implies that we can compute $\max_S \frac{|E(S)|}{|S|}$ in polynomial time.
- Saha et al in RECOMB '10 [14]prove that we can maximize the above parameter (in polynomial time) where the subset S has to contain a specific subset C of vertices. Again, this boils down to a simple computation to prove that this modified $E(\cdot)$ function is (still) supermodular.
- Aliter: this follows from general "pinching" operations on submodular/supermodular functions.
- Our viewpoint also allows us to prove (rather simply) the results for directed graph density too, thus subsuming quite a few of the results of Charikar and also of Khuller-Saha.

Examples: About $E(S)$

- Given a graph $G = (V, E)$, the function $|E(S)|$ is supermodular, while $|S|$ is modular (and thus, of course, submodular). This implies that we can compute $\max_S \frac{|E(S)|}{|S|}$ in polynomial time.
- Saha et al in RECOMB '10 [14]prove that we can maximize the above parameter (in polynomial time) where the subset S has to contain a specific subset C of vertices. Again, this boils down to a simple computation to prove that this modified $E(\cdot)$ function is (still) supermodular.
- Aliter: this follows from general "pinching" operations on submodular/supermodular functions.
- Our viewpoint also allows us to prove (rather simply) the results for directed graph density too, thus subsuming quite a few of the results of Charikar and also of Khuller-Saha.

Examples: About $E(S)$

- Given a graph $G = (V, E)$, the function $|E(S)|$ is supermodular, while $|S|$ is modular (and thus, of course, submodular). This implies that we can compute $\max_S \frac{|E(S)|}{|S|}$ in polynomial time.
- Saha et al in RECOMB '10 [14]prove that we can maximize the above parameter (in polynomial time) where the subset S has to contain a specific subset C of vertices. Again, this boils down to a simple computation to prove that this modified $E(\cdot)$ function is (still) supermodular.
- Aliter: this follows from general “pinching" operations on submodular/supermodular functions.
- Our viewpoint also allows us to prove (rather simply) the results for directed graph density too, thus subsuming quite a few of the results of Charikar and also of Khuller-Saha.

Examples: About $E(S)$

- Given a graph $G = (V, E)$, the function $|E(S)|$ is supermodular, while $|S|$ is modular (and thus, of course, submodular). This implies that we can compute $\max_S \dfrac{|E(S)|}{|S|}$ in polynomial time.
- Saha et al in RECOMB '10 [14]prove that we can maximize the above parameter (in polynomial time) where the subset S has to contain a specific subset C of vertices. Again, this boils down to a simple computation to prove that this modified $E(\cdot)$ function is (still) supermodular.
- Aliter: this follows from general “pinching" operations on submodular/supermodular functions.
- Our viewpoint also allows us to prove (rather simply) the results for directed graph density too, thus subsuming quite a few of the results of Charikar and also of Khuller-Saha.

More about graph density

- Consider the following (modified) graph density problem: we want the most dense subgraph among those that have *at least* k vertices.
- Formally, we are trying to compute the parameter $\max_{S:|S|\geq k} |E(S)|/|S|$.
- This parameter is NP-hard. Andersen shows that we can approximate this within a factor of 2. Khuller and Saha show faster implementations of the 2-factor approximation.
- A generalization: Andersen's result holds for supermodular, monotone f ($E(S)$ is one instantiation).
- Open Question: How about the problem for nonmonotone f?

More about graph density

- On the other hand, constrain the graph density parameter to consider sets S that have *at most* k vertices.
- This problem becomes very hard: Khot shows that there cannot be any PTAS for this problem unless $P = NP$.

Examples – More samplers – About $\delta(S)$

- Given a graph $G = (V, E)$, the function $\delta(S)$ is submodular. Thus we can compute $\min_S \frac{\delta(S)}{|S|}$ in polynomial time.
- And numerous other such results. For instance, we may compute the above parameter even when the subset S is *constrained* to contain a specific subset C of vertices. Or if the subset S is always constrained to have an *odd* number of vertices. Etc. Etc.

Examples – More samplers – About $\delta(S)$

- Given a graph $G = (V, E)$, the function $\delta(S)$ is submodular. Thus we can compute $\min_S \frac{\delta(S)}{|S|}$ in polynomial time.
- And numerous other such results. For instance, we may compute the above parameter even when the subset S is *constrained* to contain a specific subset C of vertices. Or if the subset S is always constrained to have an *odd* number of vertices. Etc. Etc.

Summary

- Given any arbitrary optimization problem, first check if the related function is submodular/supermodular.
- Check the other constraints. Are they matroid constraints? Are they knapsack constraints?
- Correspondingly proceed with the optimization.

- Open Questions
  - Minimize submodular functions faster.
  - Minimize submodular functions to better than 2-factors for the problem with cardinality/matroid/knapsack constraints. Done by Goemans and Soto
  - Maximize arbitrary submodular functions to better than the factor of 0.41.

References I

Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák.
Maximizing a submodular set function subject to a matroid constraint (extended abstract).
In *IPCO*, pages 182–196, 2007.

Moses Charikar.
Greedy approximation algorithms for finding dense components in a graph.
In *APPROX*, pages 84–95, 2000.

Chandra Chekuri and Jan Vondrák.
Randomized pipage rounding for matroid polytopes and applications.
CoRR, abs/0909.4348, 2009.

References II

- Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák.
  Maximizing non-monotone submodular functions.
  In *FOCS*, pages 461–471, 2007.
- L. R. Ford and D. R. Fulkerson.
  Flows in Networks.
  Princeton University Press, 1962.
- Satoru Fujishige.
  Submodular Functions and Optimization.
  Annals of Discrete Mathematics. Elsevier, second edition, 2005.
- A. V. Goldberg.
  Finding a maximum density subgraph.
  Technical report, UC Berkeley, 1984.

References III

- Martin Grötschel, László Lovász, and Alexander Schrijver.
  The ellipsoid method and its consequences in combinatorial optimization.
  Combinatorica, 1(2):169–197, 1981.

- R. Kannan and V. Vinay.
  Analyzing the structure of large graphs.
  Technical report, 1999.

- Samir Khuller and Barna Saha.
  On finding dense subgraphs.
  In *ICALP (1)*, pages 597–608, 2009.

- Jon Lee, Maxim Sviridenko, and Jan Vondrák.
  Submodular maximization over multiple matroids via generalized exchange properties.
  In *APPROX-RANDOM*, pages 244–257, 2009.

References IV

Jon Lee, Maxim Sviridenko, and Jan Vondrák.
Matroid matching: the power of local search.
In *STOC*, pages 369–378, 2010.

H. Narayanan.
Submodular Functions and Electrical Networks.
Annals of Discrete Mathematics. Elsevier, first edition, 1997.

Barna Saha, Allison Hoch, Samir Khuller, Louiqa Raschid, and Xiao-Ning Zhang.
Dense subgraphs with restrictions and applications to gene annotation graphs.
In *RECOMB*, pages 456–472, 2010.

Alexander Schrijver.
A combinatorial algorithm minimizing submodular functions in strongly polynomial time.
J. Comb. Theory, Ser. B, 80(2):346–355, 2000.