Submodular functions

Yuri Faenza

IEOR, Columbia University

yf2414@columbia.edu

Last updated on May, 09th 2021

Abstract

These notes contain examples of submodular functions and describe certain algorithms for optimizing them. They are intended for students from the M.Sc. class IEORE4008 - *Computational Discrete Optimization* and the Ph.D. class IEORE6614 - *Optimization II* in the IEOR Department at Columbia University. While better algorithms (in terms of running time and/or quality of the output solution) are often known, those discussed in here are accessible to Ph.D. students (and, in some cases, to M.Sc. students) with a general background in discrete optimization. I would be glad if you could report to me any error or typo you find in these notes.

Contents

1 Definition, basic properties, and examples

Let V be a finite set, and denote by 2^V the *power set* of V, i.e., the family of all subsets of V. A function $f : 2^V \to \mathbb{R}$ is called *submodular* if, for each $A, B \in 2^V$, we have:

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B). \tag{1}$$

Let us start by proving an equivalent definition of submodularity.

Lemma 1. *Let $f : 2^V \to \mathbb{R}$ be a function. f is submodular if and only if it satisfies the* law of diminishing returns*, i.e.*

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B) \quad \text{for all } A \subseteq B \in 2^V \text{ and } e \in V \setminus B. \tag{2}$$

Proof. Assume f is submodular, and let A, B, e be as in (2). Define $A' := A \cup \{e\}$ and $B' := B$. Then from (1), we have:

$$f(A \cup \{e\}) + f(B) = f(A') + f(B') \geq f(A' \cap B') + f(A' \cup B') = f(A) + f(B \cup \{e\}),$$

as required.

Now assume f satisfies (2). Let $B \setminus A = \{e_1, \ldots, e_k\}$. Then, by repeatedly applying (2), we have:

$$\begin{aligned} f(A \cap B) - f(A) &\leq f((A \cap B) \cup \{e_1\}) - f(A \cup \{e_1\}) \\ &\leq f((A \cap B) \cup \{e_1, e_2\}) - f(A \cup \{e_1, e_2\}) \\ &\leq \ldots \\ &\leq f(B) - f(A \cup B), \end{aligned}$$

as required. □

Next exercise shows that the previous definition can be simplified to sets T consisting of only one element more than S.

Exercise 1. Let $f : 2^V \to \mathbb{R}$ be a function. f is submodular if and only if for each $S \subseteq V$ and $i, j \in V \setminus S$, we have:

$$f(S \cup \{i\}) - f(S) \geq f(S \cup \{i, j\}) - f(S \cup \{j\}). \tag{3}$$

As next lemmas show, submodularity is preserved when taking nonnegative combinations and complement functions.

Lemma 2. *Let $f, g : 2^V \to \mathbb{R}$ be submodular functions, and $\lambda, \alpha \geq 0$. Then $\lambda f + \alpha g$ is a submodular function.*

Proof. Immediately from the definition. □

Exercise 2. Let $f : 2^V \to \mathbb{R}$ be a submodular function. Show that the function $g : 2^V \to \mathbb{R}$ defined as follows:

$$g(A) := f(V \setminus A) \text{ for all } A \subseteq V$$

is submodular.

Many submodular functions enjoy additional properties. We say that a submodular function $f : 2^V \to \mathbb{R}$ is *monotone* if $A \subseteq B \subseteq V$ implies $f(A) \leq f(B)$. We say it is *symmetric* if $f(A) = f(V \setminus A)$ for all $A \subseteq V$. Lemma 2 and Exercise 2 imply the following.

Lemma 3. *If $f : 2^V \to \mathbb{R}$ is a submodular function, then the function $h : 2^V \to \mathbb{R}$ defined as $h(A) := f(A) + f(V \setminus A)$ for all $A \subseteq V$ is also submodular.*

Another useful property is the following.

Lemma 4. *Let $f : 2^V \to \mathbb{R}_{\geq 0}$ be a submodular function, $S \subseteq T$. Then*

$$f(T) \leq f(S) + \sum_{e \in T \setminus S} (f(S \cup \{e\}) - f(S)). \tag{4}$$

If f is also monotone, then (4) also holds if $S \not\subseteq T$.

Proof. Let $T \setminus S = \{e_1, \ldots, e_k\}$.

$$\begin{aligned} f(T) - f(S) &= \textstyle\sum_{i=1}^{k} (f(S \cup \{e_1, \ldots, e_i\}) - f(S \cup \{e_1, \ldots, e_{i-1}\})) && \text{(telescopic sum)} \\ &\leq \textstyle\sum_{i=1}^{k} (f(S \cup \{e_i\}) - f(S)) && \text{(by submodularity).} \end{aligned}$$

If moreover f is monotone and $S \not\subseteq T$, we have $f(T) - f(S) \leq f(T \cup S) - f(S)$, and the statement follows from the first part. □

1.1 Examples

Submodular functions appear broadly in problems in machine learning and optimization. Let us see some examples.

Exercise 3 (Cut function)**.** Let $G(V, E)$ be a graph with a weight function $w : E \to \mathbb{R}_+$. Show that the function that associates to each set $A \subseteq V$ the value $w(\delta(A))$ is submodular.

Exercise 4. Let $G(V, E)$ be a graph. For $F \subseteq E$, define:

$$f(F) := |\{u \in V : \delta(u) \cap F \neq \emptyset\}|.$$

Show that f is a submodular function.

Example 1 (Joint entropy)**.** Consider a discrete random variable X, with support S and probability mass function (pmf) P. The *entropy* of X is given by

$$H(X) := -\sum_{x \in S} P(x) \log(P(x)),$$

where the basis of the logarithm function is 2 and we assume $P(x) \log(P(x)) = 0$ when $P(x) = 0$. The entropy gives a measures of how "unpredictable" a random variable is. For instance, take X to be a (possibly biased) coin, and define:

$$P(X) = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1 - p \end{cases}$$

For $p = .5$, i.e., the coin is unbiased, we have $H(X) = 1$, while for $p \in \{0, 1\}$ we have $H(X) = 0$.

If we are given a finite set of random variables $X_1, \ldots, X_n$ with supports $S_1, \ldots, S_n$, their *joint entropy* is given by:

$$H(X_1, \ldots, X_n) = -\sum_{x_1 \in S_1} \sum_{x_2 \in S_2} \ldots \sum_{x_p \in S_n} P(x_1, \ldots, x_n) \log(P(x_1, \ldots, x_n)),$$

where $P(x_1, \ldots, x_n)$ is the joint pmf and we let $H(\emptyset) = 0$. Note that $H(\mathcal{X}) \geq 0$ for all $\mathcal{X} \subseteq \{X_1, \ldots, X_n\}$, since it is the summation of a finite number of nonnegative terms. Given families $\mathcal{X}, \mathcal{Y} \subseteq \{X_1, \ldots, X_n\}$, we define the *conditional entropy*

$$H(\mathcal{X}|\mathcal{Y}) := H(\mathcal{Y} \cup \mathcal{X}) - H(\mathcal{Y}).$$

One can verify that $H(\mathcal{X}|\mathcal{Y}) \leq H(\mathcal{X}|\mathcal{Y}')$ for $\mathcal{Y}' \subseteq \mathcal{Y}$, where we define $H(\mathcal{X}|\emptyset) = H(\mathcal{X})$ - i.e., the larger the set of variables we condition on, the smaller is the conditional entropy. We claim that the joint entropy function is submodular. We show this via (2). Let $\mathcal{X} \subseteq \mathcal{Y} \subseteq \{X_1, \ldots, X_n\}$, and $X \in \{X_1, \ldots, X_n\} \setminus \mathcal{Y}$. Then:

$$H(\mathcal{X} \cup \{X\}) - H(\mathcal{X}) = H(\{X\}|\mathcal{X}) \geq H(\{X\}|\mathcal{Y}) = H(\mathcal{Y} \cup \{X\}) - H(\mathcal{Y}).$$

Example 2 (Symmetric Mutual information)**.** Consider again the joint entropy defined in Example 1 and the notation introduced there. Abbreviate $\mathcal{Z} := \{X_1, \ldots, X_n\}$. Given sets $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{Z}$, define their *mutual information* as:

$$I(\mathcal{X}, \mathcal{Y}) := H(\mathcal{X}) - H(\mathcal{X}|\mathcal{Y}) = H(\mathcal{X}) + H(\mathcal{Y}) - H(\mathcal{X} \cup \mathcal{Y}).$$

Intuitively, the mutual information compute how much the entropy of $\mathcal{X}$ decreases when we condition on $\mathcal{Y}$. Since conditioning does not increase the entropy, we have

$$I(\mathcal{X}, \mathcal{Y}) = H(\mathcal{Y}) - H(\mathcal{Y}|\mathcal{X}) \geq 0.$$

Now define the *symmetric mutual information* function as follows:

$$I(\mathcal{X}) := I(\mathcal{X}, \mathcal{Z} \setminus \mathcal{X}) = H(\mathcal{X}) + H(\mathcal{Z} \setminus \mathcal{X}) - H(\mathcal{Z}).$$

Intuitively, if $I(\mathcal{X})$ is big, then seeing a realization of variables in $\mathcal{X}$ highly reduces the entropy of variables in $\mathcal{Z} \setminus \mathcal{X}$. Letting $\mathcal{X} \subseteq \mathcal{Y} \subseteq \mathcal{Z}$, and $X \in \mathcal{Z} \setminus \mathcal{Y}$, we have:

$$\begin{aligned} I(\mathcal{X} \cup \{X\}) - I(\mathcal{X}) &= H(\mathcal{X} \cup \{X\}) + H(\mathcal{Z} \setminus (\mathcal{X} \cup \{X\})) - H(\mathcal{Z}) - H(\mathcal{X}) - H(\mathcal{Z} \setminus \mathcal{X}) + H(\mathcal{Z}) \\ &= H(\mathcal{X} \cup \{X\}) + H(\mathcal{Z} \setminus (\mathcal{X} \cup \{X\})) - H(\mathcal{X}) - H(\mathcal{Z} \setminus \mathcal{X}) \end{aligned}$$

and similarly

$$I(\mathcal{Y} \cup \{X\}) - I(\mathcal{Y}) = H(\mathcal{Y} \cup \{X\}) + H(\mathcal{Z} \setminus (\mathcal{Y} \cup \{X\})) - H(\mathcal{Y}) - H(\mathcal{Z} \setminus \mathcal{Y}) \ .$$

We have therefore

$$\begin{aligned} I(\mathcal{X} \cup \{X\}) - I(\mathcal{X}) &= (H(\mathcal{X} \cup \{X\}) - H(\mathcal{X})) + (H(\mathcal{Z} \setminus (\mathcal{X} \cup \{X\})) - H(\mathcal{Z} \setminus \mathcal{X})) \\ &\geq (H(\mathcal{Y} \cup \{X\}) - H(\mathcal{Y})) + (H(\mathcal{Z} \setminus (\mathcal{Y} \cup \{X\})) - H(\mathcal{Z} \setminus \mathcal{Y})) \\ &= I(\mathcal{Y} \cup \{X\}) - I(\mathcal{Y}), \end{aligned}$$

where we used that H is submodular hence, by Lemma 2, its complement function is also submodular. Hence, the symmetric mutual information is submodular.

Example 3 (Sensor placement)**.** Consider the following (vague) problem: we have to place sensors in a river in such a way to obtain as much information as possible on the status of the river (could be the temperature, the pollution level, etc.). Locations for sensors must chosen from a set V. This problem can be made precise by formulating it using submodular functions.

As a first example, assume that, by placing a sensor in location $v \in V$, an area of radius r_v will be covered. Hence define the function $f : 2^V \to \mathbb{R}_+$ as follows:

$$f(S) = A(\cup_{v \in S} C_v) \text{ for } S \subseteq V,$$

where C_v is the circle of radius r_v centered at v for each $v \in V$, and $A(C)$ is the area of set C. $f(S)$ measures therefore the area covered by sensors placed in locations from S. Using (2), one immediately checks that f is submodular.

As a second example, we can model the problem using the symmetric mutual information from Example 2. Indeed, we can associate to each location $v \in V$, a random variable X_v, which models the measurement in the area around location v. Placing a sensor in location v will realize the random variable X_v. However, placing a sensor in a location close to v can reduce the entropy of X_v (e.g., the temperature in a location is probably close to the one few meters away), while placing it in a location much further away will not. For appropriately defined random variables $\mathcal{Z} = \{X_1, \ldots, X_{|V|}\}$, and for $\mathcal{X} \subseteq \mathcal{Z}$ we can therefore define $h(\mathcal{X}) := I(\mathcal{X})$ as a measure of how effective placing sensors in locations X is. h is submodular by Example 2.

Example 4. (1-product of matrices). For matrices M_1, M_2, we let $M_1 \otimes M_2$ be the matrix whose set of columns is the cartesian product of the set of columns of M_1, M_2. For instance,

$$\begin{pmatrix} 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \qquad \begin{pmatrix} 1 & 0 \\ 2 & 3 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 2 & 2 & 3 & 3 & 3 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

We say that a matrix M is a 1-*product* of M_1, M_2 if it can be obtained from $M_1 \otimes M_2$ by switching rows and columns. Mutual information (see Example 2) and the minimization of symmetric submodular functions (see Section 3.1) can be employed to devise an algorithm that decides whether a given matrix M is the 1-product of two matrices.

Let our input matrix M have m rows and n columns. Let $C := (C_1, \ldots, C_m)$ be a uniformly chosen random column of M. That is, $P(C = c) = \mu(c)/n$, where $\mu(c)$ denotes the number of occurrences in M of the column c of M. For $X \subseteq [m]$ we let $C_X := (C_i)_{i \in X}$ and $C_{\overline{X}} := (C_i)_{i \in \overline{X}}$, where $\overline{X} := [m] \setminus X$. Define $f : 2^{[m]} \to \mathbb{R}$ to be the mutual information of C_X and $C_{\overline{X}}$ – that is, for $X \subseteq [m]$, we let $f(X) := I(C_X; C_{\overline{X}})$. We already observed in Example 2 that f is nonnegative and submodular. Note also that f is symmetric and that $f(X) = 0$ if and only if $C_X, C_{\overline{X}}$ are independent random variables.

We next argue that M is a 1-product of some matrices M_1, M_2 if and only if $\min_{\emptyset \subsetneq X \subsetneq V} f(X) = 0$. The latter can be checked using the algorithm from Section 3.1.

First, we prove the "only if" direction. Suppose that M is the 1-product of two matrices M_1, M_2, with n_1, n_2 columns respectively. Then each row of M corresponds either to a row of M_1, or to a row of M_2. Let X be the collection of rows of M that correspond to a row of M_1.

Any column c of M can be written as $c = (c_X, c_{\overline{X}})$, where c_X (resp. $c_{\overline{X}}$) is the subvector of c corresponding to rows in X (resp. $\overline{X}$). We have $\mu(c) = \mu_1(c_X)\mu_2(c_{\overline{X}})$, where μ_i denotes the

multiplicity of a column in M_i, $i = 1, 2$. Hence

$$P(C_X = c_X,\ C_{\overline{X}} = c_{\overline{X}}) = P(C = c) = \frac{\mu(c)}{n} = \frac{\mu_1(c_X) \cdot n_2}{n} \cdot \frac{n_1 \cdot \mu_2(c_{\overline{X}})}{n} = P(C_X = c_X)(C_{\overline{X}} = c_{\overline{X}}),$$

where we used $n = n_1 n_2$. This proves that C_X and $C_{\overline{X}}$ are independent, hence $f(X) = 0$.

Exercise 5. Prove the "if" direction.

Exercise 6 (Common genetic information)**.** Consider a set of species $\mathcal{S} := \{S_1, \ldots, S_n\}$, each represented by its DNA sequence, which can be thought of as a ordered string of the entries $\{A, T, C, G\}$ (with repetitions). For instance, we could have:

$$S_1 = ATTCCGCGCGC,\ S_2 = ACGGGCTACTC,\ \ldots\ ,\ S_n = ACCCCATGCAG.$$

For $S \in \mathcal{S}$ and an ordered string s, we say that s *appears as a substring in* S if entries of s appears in S in the same order they appear in s (possibly non-consecutively). Given $S, S' \in \mathcal{S}$, define their *common genetic information* as

$$I_{CG}(S, S') = |\{s : s \text{ is a substring that appears in } S_1\} \cap \{s : s \text{ is a substring that appears in } S_2\}|.$$

The common genetic information is a measure of how close the DNAs of two species are. For $\mathcal{X} \subseteq \mathcal{S}$, define the *symmetric common genetic information* as:

$$I_{CG}(\mathcal{X}) := \\ |\{\cup_{S_1 \in \mathcal{X}, S_2 \in \mathcal{S} \setminus \mathcal{X}} \{s : s \text{ is a substring that appears in } S_1\} \\ \cap \\ \{s : s \text{ is a substring that appears in } S_2\}\}|.$$

The symmetric common genetic information is a measure of how close the DNAs of species across different sets of the partition are. Show that the symmetric common genetic information is submodular.

Example 5 (Set Covering)**.** Suppose we have a set of objects V, and let $S_1, \ldots, S_n$ be a number of (possibly overlapping) subsets of V. Given $S \subseteq V$, let $f(S)$ be the number of sets from $S_1, \ldots, S_n$ that have non-empty intersections with S. Using (2), one easily checks that f is submodular.

Example 6 (Influence maximization in social networks)**.** Let $G(V, E)$ be a graph, modeling a (social) network. Suppose we would like to measure as an information spreads in the network. To each arc e of the graph we associate a probability $p(e)$. The spreading of the information goes as follows. At round 0, a set $S \subseteq V$ is communicated the information. At each round $i \geq 1$, independently for each node u that has been communicated the information for the first time in round $i - 1$ and for each edge $uv \in E$, e is *realized* with probability $p(e)$, i.e. v will be communicated the information with probability $p(e)$ (note that v may already have been communicated the information, in which case the outcome is not important). We stop when we reach a round when there is no new person that has been communicated the information. For $S \subseteq V$, we let $f(S)$ be the expected number of people that are communicated the information, assuming that in round 0 the information was communicated to S.

We show that f is submodular. This follows from a simple observation: wlog we assume that whether any arc $e = uv$ is realized or not is computed a priori, i.e., at the very beginning of the

procedure. If none of the endpoints of e was communicated the information during the procedure, it does not matter whether e is realized or not. If conversely one of u, v is communicated the information and e is realized, then both u and v are eventually communicated the information.

Let c be the outcome of a sampling of all arcs of G, and define $f_c(S)$ to be the number of people that are communicated the information, assuming that in round 0 the information was communicated to S and that the realization of arcs is exactly as in c. Observe that $f(S) = \sum_c p(c) f_c(S)$, where $p(c)$ is the probability of outcome c and the sum is over all possible c. By Lemma 2, it suffices to show that $f_c(S)$ is submodular. Observe that, if G^c is the subgraph G obtained by keeping only the arcs realized in c, then a node v is communicated the information if and only if there exists a node in S that belongs to the same connected component of v in G^c. Hence $f_c(S)$ counts the number of connected components of G^c that intersect S. This is an instance of set covering, see Example 5, hence f_c is submodular. Note that computing $f_c(S)$ for a fixed c is NP-Hard [7].

Example 7 (Document summarization)**.** Given a collection of documents or images N, we want to measure how much a set $S \subseteq N$ "represents" the original collection N. In order to model this problem, consider for instance, the problem solved by a search engine when a query arrives. On the one hand, it wants to show content that is most relevant for the search. On the other hand, it does not want to show two very similar documents. We can model the quality of the set S as

$$f(S) = R(S) + D(S),$$

where $R(S)$ measures the *relevance* and $D(S)$ measures the *diversity*.

A typical choice for $R(S)$ is known as the *facility location function*:

$$R(S) = \sum_{a \in N} \left\{ \max_{b \in S} s(a, b) \right\},$$

where $s(a, b)$ is a *similarity* measure between a and b. A typical choice for $D(S)$ is

$$D(S) = \sum_j \sqrt{|S \cap P_j|},$$

where $\{P_j\}_j$ is a partition of N. It can be easily shown that both R and D are submodular functions.

Example 8 (Rank function of a matroid)**.** Let V be a finite set and $\mathcal{I}$ a family of subsets of V. $(V, \mathcal{I})$ is a *matroid* if it satisfies the following three properties:

(M0) $\emptyset \in \mathcal{I}$;

(M1) $J \subseteq J' \in \mathcal{I} \Rightarrow J \in \mathcal{I}$;

(M2) Let $S \subseteq V$, and J, J' be inclusionwise maximal subsets of S such that $J, J' \in \mathcal{I}$. Then $|J| = |J'|$.

The *rank function* of a matroid is the function $r : 2^V \to \mathbb{N} \cup \{0\}$ defined as follows:

$$r(S) := \max\{|J| : J \subseteq S, J \in \mathcal{I}\}. \tag{5}$$

Note that r is well-defined because of (M2). For a given set S, a set J achieving (5) is called a *basis of* S. A set achieving $r(V)$ is called a *basis of the matroid*.

We now show that r is submodular, by proving that, for every $A, B \subseteq V$, (1) holds. Let $J_\cap$ be a basis of $A \cap B$. Note that we can "expand" $J_\cap$ to a basis of A – that is, create a set $J_A \supseteq J_\cap$ that is a basis of A. Indeed, we can let J_A be any inclusionwise maximal set from $\mathcal{I}$ such that $J_\cap \subseteq J_A \subseteq A$. Then, by (M2), J_A realizes $r(A)$. Observe moreover that

$$J_A \cap B = J_\cap. \tag{6}$$

Indeed, by construction, $J_A \cap B \supseteq J_\cap$. Moreover, by (M1), $J_A \cap B \in \mathcal{I}$, and $J_A \cap B \supsetneq J_\cap$ contradicts the fact that $J_\cap$ is a basis of $A \cap B$.

Similarly, we can "expand" J_A to a basis $J_\cup$ of $A \cup B$, that is

$$J_\cup \cap A = J_A. \tag{7}$$

We conclude therefore

$$r(A) + \underbrace{r(B)}_{\geq |J_\cup| - |J_A| + |J_\cap|} - r(A \cup B) - r(A \cap B) \geq |J_A| + |J_\cup| - |J_A| + |J_\cap| - |J_\cup| - |J_\cap| \geq 0,$$

where we used that, by (6) and (7), $J_B := J_\cup \setminus (J_A \setminus J_\cap) \subseteq B$ and $|J_B| = |J_\cup \setminus (J_A \setminus J_\cap)| = |J_\cup| - |J_A| + |J_\cap|$. Since, by (M1), $J_B \in \mathcal{I}$, we have $r(B) \geq |J_B|$. Rearranging gives the thesis.

When dealing with algorithms that involve submodular functions, we will always assume that *evaluating* the function requires constant time. This may not be true, and in cases where it is not, then the running time of the algorithms must be appropriately scaled. We call each evaluation of the function an *oracle call*.

1.2 Measuring how "close to submodular" a nonnegative function is

While submodularity is a "yes/no" concept – that is, a function is either submodular or it is not, we can define a continuous parameter describing how "close to submodular" is a function that is not submodular. Let $f : 2^V \to \mathbb{Z}_{>0}$ be monotone. The *submodularity ratio* of f is

$$q(f) := \min_{S \subseteq T \subseteq V} \frac{\sum_{x \in T \setminus S} (f(S \cup \{x\}) - f(S))}{f(T) - f(S)},$$

where we define $0/0 := 1$.

Lemma 5. *Let $f : 2^V \to \mathbb{Z}_{>0}$ be monotone. Then f is submodular if and only if $q(f) \geq 1$.*

Proof. Suppose f is submodular and let $S \subseteq T \subseteq V$. Applying Lemma 4, we deduce

$$f(T) \leq f(S) + \sum_{x \in T \setminus S} (f(S \cup \{x\}) - f(S)),$$

and the statement follows. Conversely, let f not be submodular. Applying Exercise 1, there exist $S \subseteq V$, and elements $i, j \in V \setminus S$ such that

$$f(S \cup \{i\}) - f(S) < f(S \cup \{i, j\}) - f(S \cup \{j\}). \tag{8}$$

Taking $S = S$, $T = S \cup \{i, j\}$ in the formula of $q(f)$, we have

$$\begin{aligned} q(f) &\leq \frac{f(S\cup\{i\})-f(S)+f(S\cup\{j\})-f(S)}{f(S\cup\{i,j\})-f(S)} \\ &= \frac{f(S\cup\{i\})-f(S)+f(S\cup\{j\})-f(S)}{f(S\cup\{i,j\})-f(S\cup\{j\})+f(S\cup\{j\})-f(S)} \\ &< \frac{f(S\cup\{i\})-f(S)+f(S\cup\{j\})-f(S)}{f(S\cup\{i\})-f(S)+f(S\cup\{j\})-f(S)} \\ &= 1. \end{aligned}$$

where in the strict inequality we used (8). □

Often, algorithms for monotone submodular functions can be employed for general monotone functions, with the quality we can guarantee for the output degrading as $q(f)$ becomes smaller. See Section 2.1.1.

2 Maximizing a submodular function

In this section, we deal with a number of problems that aim at maximizing submodular functions. As we will see, all such problems are NP-Hard. Hence, the goal will be to obtain approximation algorithms.

It makes sense to assume throughout the section that $f : 2^V \to \mathbb{R}_+$, else no approximation factor can be guaranteed. Indeed, let S^* be an optimal solution to, say, the problem of maximizing a submodular function f over a ground set V. Now define the function $f'(S) = f(S) - f(S^*)$. Clearly, the maximum of f' is achieved at S^*, and has value 0, while all non-optimal sets S satisfy $f'(S) < 0$. Using (1), one easily verifies that f' is submodular. Hence, for any non-optimal set S we have $f'(S^*)/f(S) = 0$. So any constant factor approximation to the problem of maximizing a submodular function needs to find the maximum of f', contradicting its NP-Hardness (or the corresponding complexity theory assumption).

When considering an approximation algorithm, we say it is an α-approximation if it outputs a solution at least α times the optimum (hence, $\alpha \leq 1$).

2.1 Monotone, with cardinality constraint

Consider the following problem `MAX-MC`:

Given: A monotone (i.e., $f(S) \leq f(T)$ for $S \subseteq T$) submodular function $f : 2^V \to \mathbb{R}_+$, and a number $k \in \mathbb{N}$.
Find: A set $\emptyset \subseteq X \subseteq V$ of cardinality k that maximizes $f(X)$.

Note that we could have replaced *of cardinality k* with *of cardinality at most k* without changing the problem, since we assume that the function is monotone.

Theorem 6. *`MAX-MC` is NP-Hard.*

A simple greedy algorithm gives a constant-factor approximation to `MAX-MC`.

Theorem 7. *Algorithm 1 gives a* $1 - (1 - \frac{1}{k})^k \geq 1 - \frac{1}{e}$*-approximation `MAX-MC`.*

Algorithm 1 Greedy algorithm for MAX-MC.

Require: A monotone submodular function $f : 2^V \to \mathbb{R}_+$ and $k \in \mathbb{N}$.

1: Set $S = \emptyset$.
2: **for** $i = 1, \ldots, k$ **do**
3: Select $v \in V \setminus S$ that maximizes $f(S \cup \{v\})$. Set $S = S \cup \{v\}$.
4: **end for**
5: **return** S.

Proof. Let $S^* = \{e_1, \ldots, e_k\}$ be an optimal solution and, for $\ell = 0, \ldots, k$, let S_ℓ be the solution constructed by the algorithm in iteration ℓ. We have:

$$\begin{aligned} f(S^*) - f(S_\ell) &\leq \textstyle\sum_{i=1}^{k} (f(S_\ell \cup \{e_i\}) - f(S_\ell)) && \text{(by Lemma 4)} \\ &< k(f(S_{\ell+1}) - f(S_\ell)) && \text{(by greedy procedure).} \end{aligned}$$

Hence, we have:

$$f(S_{\ell+1}) - f(S_\ell) \geq \frac{1}{k}(f(S^*) - f(S_\ell)) \;\Leftrightarrow\; f(S^*) - f(S_{\ell+1}) \leq (1 - \frac{1}{k})(f(S^*) - f(S_\ell)).$$

We conclude:

$$\begin{aligned} f(S^*) - f(S_k) &\leq (1 - \tfrac{1}{k})(f(S^*) - f(S_{k-1}) \\ &\leq (1 - \tfrac{1}{k})^2(f(S^*) - f(S_{k-2})) \\ &\leq \ldots \leq (1 - \tfrac{1}{k})^k((f(S^*) - f(S_0)) \\ &\leq (1 - \tfrac{1}{k})^k f(S^*) && \text{(by nonnegativity).} \end{aligned}$$

□

Algorithm 1 is the best possible: it can be shown that no polynomial-time algorithm can achieve an approximation ratio strictly better than $(1 - \frac{1}{e})$.

Exercise 7 (Lazy greedy)**.** Since evaluating a submodular function can be expensive and/or the ground set may be very large, the greedy algorithm for maximizing non-negative monotone submodular functions is often replaced in practice by the so-called *Lazy greedy*. Assume $f(\emptyset) = 0$, which can be achieved by replacing f with $f - f(\emptyset)$. At the first step, Lazy greedy computes for $e \in V$ the value $p(e) = f(\{e\})$, adds to S the element e_1 that maximizes $p(e_1)$, and sorts the remaining values $p(e)$ in non-increasing order. At the i-th step, instead of computing $f(S \cup \{e\}) - f(S)$ for all $e \in V \setminus S$, Lazy greedy takes the element e from $V \setminus S$ that maximizes $p(e)$, and updates $p(e) = f(S \cup \{e\}) - f(S)$. If e still maximizes $p(f)$ for $f \in V \setminus S$, then e is added to S and the algorithm proceeds to the next step. Else, let e' be the new maximizer, update $p(e') = f(S \cup \{e'\})f(S)$, again check if e' still maximizes $p(e)$, and proceed as above. Prove that Lazy greedy will not perform more evaluations of f than the classical greedy, but it still has an $(1 - \frac{1}{e})$ approximation guarantee (for the original function).

Exercise 8. Let $f : 2^V \to \mathbb{R}$ be a monotone submodular function. Show that an optimal solution to MAX-MC can be obtained by solving the following problem:

$$\begin{array}{lllll} \max & \mu & & & \\ \text{s.t.} & \mu & \leq & f(S) + \sum e \in V \setminus S x_e \cdot (f(S \cup \{e\} - f(S)) & S \subseteq V \\ & \sum_{e \in V} & \leq & k & \\ & x_e & \in & \{0,1\} & \forall e \in V. \end{array}$$

2.1.1 Some consequences of the greedy algorithm

As a first consequence, we analyze the performance of the greedy algorithm when f is not submodular, as a function of $q(f)$.

Exercise 9. Apply Algorithm 1 on input $k \in \mathbb{N}$ and a monotone function $f : 2^V \to \mathbb{R}$, and let $\overline{S}$ be set it outputs. Let S^* be a set achieving $\max_{S \subseteq V : |S| \leq k} f(S)$. Show that $f(\overline{S}) \geq (1 - e^{-q(f)}) f(S^*)$.

Now consider the following *maximum coverage* problem. We are given a monotone function[1] $f : 2^V \to \mathbb{N}$ and $t \in \mathbb{N}$, with $t \leq f(V)$. We want to find a set $S \subseteq V$ such that $f(S) \geq t$, and S is of minimum cardinality among such sets. To do that, we apply the following appropriate modification of Algorithm 1, see Algorithm 2.

Algorithm 2 Greedy algorithm for maximum coverage.

Require: A monotone function $f : 2^V \to \mathbb{N}$ and $t \in \mathbb{N}$.
1: Set $S = \emptyset$.
2: **while** $f(S) < t$ **do**
3: Select $v \in V \setminus S$ that maximizes $f(S \cup \{v\})$. Set $S = S \cup \{v\}$.
4: **end for**
5: **return** S.

Let $\overline{S}$ be the set output by the algorithm and S^* be an optimal solution to the problem.

Exercise 10. Show that $|\overline{S}| \leq 1 + \frac{\log t}{q(f)} |S^*|$.

2.2 Monotone, with knapsack constraint

We now consider MAX-MK, a common generalization of MAX-MC and of the classical knapsack problem.

Given: A monotone submodular function $f : 2^V \to \mathbb{R}$, a capacity B and, for each $e \in V$, a weight $w(e) \geq 0$.
Find: A set $\emptyset \subseteq X \subseteq V$ of total weight at most B that maximizes $f(X)$.

When dealing with the classical knapsack problem, we saw that an algorithm that greedily adds objects basing its decision only on their contribution to the objective function, without accounting for their weight, can perform very poorly. Moreover, we also saw that a solution made of one object only could reach a much higher profit than pure greedy, hence such solutions need to be enumerated separately.

[1] Notice that we are assuming that the codomain of f is the set of strictly positive integers. This is not required for the algorithm, but it is needed to conclude the bound from Exercise 10.

Given those two facts, the algorithm below is quite natural. Note that in this case, we first enumerate over all solutions with at most 2 objects, and then we apply weight-scaled greedy (as in the classical knapsack case) starting from all feasible sets of at most 3 objects.

Algorithm 3 Greedy algorithm for MAX-MK.

Require: A monotone submodular function $f : 2^V \to \mathbb{R}_+$, weights $w : V \to \mathbb{R}_+$, and capacity B.

1: Let $\overline{S}$ be, among the feasible solutions S with at most 2 objects, the one that maximizes $f(S)$.

2: **for** $Y \subseteq V$, $|Y| = 3$ **do**
3: **if** $w(Y) > B$ **then**
4: Set $S^Y = \emptyset$.
5: **else**
6: Set $S^Y = Y$, $I = V \setminus Y$.
7: **while** $I \neq \emptyset$ **do**
8: Let $\overline{e} \in \arg\max_{e \in I} \{\frac{f(S^Y \cup \{e\}) - f(S^Y)}{w(e)}\}$.
9: Set $I = I \setminus \{\overline{e}\}$.
10: **if** $w(S^Y \cup \{\overline{e}\}) \leq B$ **then**
11: Set $S^Y = S^Y \cup \{\overline{e}\}$.
12: **end if**
13: **end while**
14: **end if**
15: **end for**
16: **return** $\arg\max\{f(\overline{S}), \{f(S^Y) : Y \subseteq V, |Y| = 3\}\}$.

Algorithm 3 achieves for MAX-MK the same approximation guarantee as Algorithm 1 for MAX-MC.

Theorem 8. *Let $\hat{S}$ be the set output by Algorithm 3, and let S^* be the optimal solution. Then $f(\hat{S}) \geq (1 - \frac{1}{e}) f(S^*)$.*

In order to prove Theorem 8, we can assume that S^* has size at least 3, else the statement is trivial. Therefore, let $S^* = \{f_1, \ldots, f_k\}$, with $k \geq 3$, so that $f_i \in \arg\max_{f \in S^* \setminus S^*_{i-1}} f(S^*_{i-1} \cup \{f\})$, where $S^*_i = \{f_1, \ldots, f_i\}$. Fix now $Y = S^*_3 \subseteq S^*$, and consider the solution S^Y constructed by the algorithm. It is enough to show $f(S^Y) \geq (1 - \frac{1}{e}) f(S^*)$. For convenience, we will drop the Y superscript and add instead a subscript ℓ, denoting by S_ℓ the set S^Y after the ℓ-th repetition of Step 2. We also let e_ℓ be the selected object in the ℓ-th repetition of Step 3. For $Y \subseteq X \subseteq V$, let $g(X) := f(X) - f(Y)$. g is the translate of a monotone submodular function, and hence it is monotone submodular. In addition, we let $B' = B - w(Y)$.

Let ℓ^* be the first iteration when an item is not added into the set S_{ℓ^*}. We can assume that $e_{\ell^*} \in S^*$, else the removal of e_{ℓ^*} does not change the outcome of the algorithm (or the optimum).

Lemma 9. $g(S_{\ell^*-1} \cup \{e_{\ell^*}\}) \geq (1 - \frac{1}{e}) g(S^*)$.

Proof. Let $\ell \leq \ell^* - 1$. We have:

$$\begin{aligned} g(S^*) &\leq g(S_\ell) + \textstyle\sum_{e \in S^* \setminus S_\ell} g(S_\ell \cup \{e\}) - g(S_\ell) && \text{(by Lemma 4)} \\ &\leq g(S_\ell) + \textstyle\sum_{e \in S^* \setminus S_\ell} w(e) \frac{g(S_\ell \cup \{e_{\ell+1}\}) - g(S_\ell)}{w(e_{\ell+1})} && \text{(by greedy procedure)} \\ &= g(S_\ell) + \frac{g(S_\ell \cup \{e_{\ell+1}\}) - g(S_\ell)}{w(e_{\ell+1})} \textstyle\sum_{e \in S^* \setminus S_\ell} w(e) \\ &\leq g(S_\ell) + \frac{g(S_\ell \cup \{e_{\ell+1}\}) - g(S_\ell)}{w(e_{\ell+1})} B' && \text{(by feasibility of } S^*). \end{aligned}$$

We deduce

$$\begin{gathered} -\big(g(S_\ell \cup \{e_{\ell+1}\}) - g(S_\ell)\big) \leq -\frac{w(e_{\ell+1})}{B'}(g(S^*) - g(S_\ell)) \\ \Updownarrow \\ g(S^*) - g(S_\ell \cup \{e_{\ell+1}\}) \leq (1 - \frac{w(e_{\ell+1})}{B'})(g(S^*) - g(S_\ell)). \end{gathered}$$

Exercise 11. Conclude the proof of the lemma.

□

We now prove Theorem 8. Let $f^* := e_{\ell^*}$. From Lemma 9, we have:

$$f(S_{\ell^*-1} \cup \{f^*\}) - f(Y) \geq (1 - \frac{1}{e})(f(S^*) - f(Y)). \tag{9}$$

For $i = 1, \ldots, 3$, since $S_i^* \subseteq S_{\ell^*-1}$, we also have:

$$f(S_{\ell^*-1} \cup \{f^*\}) - f(S_{\ell^*-1}) \leq f(S_{i-1}^* \cup \{f^*\}) - f(S_{i-1}^*) \leq f(S_i^*) - f(S_{i-1}^*),$$

where the first inequality follows from submodularity and the second from the ordering of f_i and the fact that $f^* \in S^* \setminus \{f_1, f_2, f_3\}$. Summing over $i = 1, 2, 3$ gives:

$$3(f(S_{\ell^*-1} \cup \{f^*\}) - f(S_{\ell^*-1})) \leq f(S_3^*) - f(\emptyset) \leq f(S_3^*) = f(Y). \tag{10}$$

Using (9) and (10), we deduce:

$$\begin{aligned} f(S_{\ell^*-1}) &\geq f(S_{\ell^*-1} \cup \{f^*\}) - \tfrac{1}{3} f(Y) = f(S_{\ell^*-1} \cup \{f^*\}) - f(Y) + \tfrac{2}{3} f(Y) \\ &\geq (1 - \tfrac{1}{e})(f(S^*) - f(Y)) + \tfrac{2}{3} f(Y) \geq (1 - \tfrac{1}{e}) f(S^*). \end{aligned}$$

Since the algorithm outputs a superset of S_{ℓ^*-1} and f is monotone, the theorem follows.

2.3 Monotone, with matroid constraint

In Example 8, we have seen the definition of matroid $(V, \mathcal{I})$ and observed that the rank function of a matroid is submodular. Recall that the family of sets $\mathcal{I}$ satisfies a monotonicity property: $J \subseteq J' \in \mathcal{I} \Rightarrow J \in \mathcal{I}$. Hence, imposing that a set of objects selected from a ground set V belongs to $\mathcal{I}$ generalizes the cardinality constraint in a different way than the knapsack constraint. It is then reasonable to attack the following problem `MAX-MAT`

Algorithm 4 Greedy algorithm for `MAX-MT`.

Require: A monotone submodular function $f : 2^V \to \mathbb{R}_+$, and a matroid $(V, \mathcal{I})$.

Set $J = \emptyset$, $I = V$.
while $I \neq \emptyset$ **do**
  Let $\bar{e} \in \arg\max_{e \in I} f(S \cup \{e\})$.
  Set $I = I \setminus \{\bar{e}\}$.
  if $J \cup \{\bar{e}\} \in \mathcal{I}$ **then**
    Set $J = J \cup \{\bar{e}\}$.
  end if
end while
return J.

Given: A monotone submodular function $f : 2^V \to \mathbb{R}_+$, a matroid $(V, \mathcal{I})$.
Find: A set $J \in \mathcal{I}$ that maximizes $f(J)$.

by a suitable adaptation of Algorithm 1, see Algorithm 4.

Before investigating Algorithm 4, we need to explain the computational model we work with. Indeed, similarly to what we have seen for submodular functions, describing a matroid by listing all the sets in $\mathcal{I}$ may require size exponential in $|V|$. Instead, we assume we have access to an *independence oracle* that, whenever we choose a set $J \subseteq V$, will output whether $J \in \mathcal{I}$. In this model, both our input size and the running time of Algorithm 4 are polynomial in $|V|$.

Recall that a *basis* of a matroid $(V, \mathcal{I})$ is a set $J \in \mathcal{I}$ of maximum cardinality. We will use the following fact.

Lemma 10. *Let J, J' be two basis of $(V, \mathcal{I})$. Then $|J| = |J'|$ and there exists a bijection $\phi : B_1 \setminus B_2 \to B_2 \setminus B_1$ such that, for all $x \in B_1 \setminus B_2$, we have that $B_1 \setminus \{x\} \cup \{\phi(x)\}$ is a basis of $(V, \mathcal{I})$.*

Theorem 11. *Algorithm 4 outputs a basis J of $(V, \mathcal{I})$ with $f(J) \geq \frac{1}{2} f(J^*)$, where J^* is an optimal solution to `MAX-MAT`.*

Proof. Assume $e_1, e_2, \cdots, e_k$ are the elements of J, added in this exact order. Note that both J and J^* are bases of $(V, \mathcal{I})$ and thus we can let ϕ be a bijection from J to J^*, such that restricted to $J \setminus J^*$, ϕ satisfies Lemma 10 and restricted to $J \cap J^*$, ϕ is the identity function. Let $|J| = |J^*| =: k$, $e_i^* := \phi(e_i)$ and $J_i := \{e_1, e_2, \cdots, e_i\}$ for all $i \in [k]$ with $J_0 = \emptyset$. Then,

$$
\begin{aligned}
f(J^*) - f(J) &\leq \textstyle\sum_{e \in J^* \setminus J} f(J \cup \{e\}) - f(J) && \text{(by Lemma 4)} \\
&= \textstyle\sum_{i=1}^{k} f(J \cup \{e_i^*\}) - f(J) \\
&\leq \textstyle\sum_{i=1}^{k} f(J_{i-1} \cup \{e_i^*\}) - f(J_{i-1}) && \text{(by Lemma 1)} \\
&\leq \textstyle\sum_{i=1}^{k} f(J_{i-1} \cup \{e_i\}) - f(J_{i-1}) && \text{(because of the greedy procedure and Lemma 10)} \\
&= f(J) - f(\emptyset) && \text{(telescopic sum)} \\
&\leq f(J) && \text{(by nonnegativity of } f\text{)}.
\end{aligned}
$$

□

2.4 Non-monotone

We now consider the problem of maximizing a non-monotone submodular function, which we denote by `MAX-C`.

Given: A submodular function $f : 2^V \to \mathbb{R}$.
Find: A set $\emptyset \subseteq X \subseteq V$ that maximizes $f(X)$.

Note that `MAX-C` includes max-cut as a special case. Hence, the following holds.

Theorem 12. *`MAX-C` is NP-Hard.*

One could think of using the vanilla greedy algorithm seen for `MAX-MC`. However, this can go arbitrarily bad. Consider in fact the following example: $f : \{v_1, \dots, v_k\} \to \mathbb{R}_{\geq 0}$ defined as $f(S) = |S|$ if $v_1 \notin S$, and $f(S) = 2$ otherwise. It can be verified that the function is submodular, greedy outputs a set containing v_1, while the optimal solution is $\{v_2, \dots, v_k\}$. Understanding why greedy performs poorly on this instance will allow us to understand how to design a better algorithm.

Greedy iteratively adds the item that looks most promising at the current stage. However, it overlooks the fact that *removing* an item may be more promising than adding it. In the example above, we have:

$$f(\{v_1\}) - f(\emptyset) = 2 \qquad \text{but} \qquad f(\{v_1, \dots, v_k\} \setminus \{v_1\}) - f(\{v_1, \dots, v_k\}) = k - 2.$$

So removing v_1 from the full set looks more promising than adding it to the empty set. How reliable are those estimations? By Lemma 1, we know that:

$$f(\{v_1\}) - f(\emptyset) \geq f(S \cup \{v_1\}) \setminus f(S) \geq f(T \cup \{v_1\}) \setminus f(T) \qquad \text{for all } S \subseteq T \text{ such that } v_1 \notin T.$$

From the previous inequalities, we learn two facts: first, adding $\{v_1\}$ never gives an increase in the profit larger than 2. Second, for a more accurate estimate of the effect of adding v_1 to the output solution T, we can compute $f(S \cup \{v_1\}) \setminus f(S)$ for a set S that is contained in T, and as similar as possible to it. One the other hand, by Exercise 2, $g : \{v_1, \dots, v_k\} \to \mathbb{R}_{\geq 0}$ defined as $g(S) := f(\{v_1, \dots, v_k\} \setminus S)$ for $S \subseteq \{v_1, \dots, v_k\}$ is also submodular, hence similarly

$$f(T \setminus \{v_1\}) - f(T) \geq f(S) - f(S \cup \{v_1\}) \geq f(\emptyset) - f(\{v_1\}) \qquad \text{for all } S \subseteq T \text{ such that } v_1 \in T \setminus S.$$

holds. Following this intuition, we consider therefore the following algorithm, called *double greedy*.

Theorem 13. *Double greedy is a $\frac{1}{3}$-approximation to `MAX-C`.*

Proof. In order to prove Theorem 13, we start with the observation that not both adding and removing any of the e_i can decrease the value of the function. Actually, we show a stronger statement.

Exercise 12. In each step of Algorithm 5, $a + b \geq 0$.

Now let OPT be any optimal solution to `MAX-C`, and consider the family of solutions $OPT_0, \dots, OPT_n$, defined as follows: for $i \in \{0, \dots, n\}$, set

$$OPT_i = (Y_i \cap \{e_1, \dots, e_i\}) \cup (OPT \cap \{e_{i+1}, \dots, e_n\}).$$

Algorithm 5 Double greedy algorithm for `MAX-C`.

Require: A submodular function $f : 2^V \to \mathbb{R}_+$, with $V = \{e_1, \ldots, e_n\}$.

Let $X_0 = \emptyset$, $Y_0 = V$.
for $i = 1, \ldots, n$ **do**
  Let $a = f(X_{i-1} \cup \{e_i\}) - f(X_{i-1})$, $b = f(Y_{i-1} \setminus \{e_i\}) - f(Y_{i-1})$.
  if $a \geq b$ **then**
    Set $X_i = X_{i-1} \cup \{e_i\}$, $Y_i = Y_{i-1}$.
  else
    Set $X_i = X_{i-1}$, $Y_i = Y_{i-1} \setminus \{e_i\}$.
  end if
end for
return X_n (or, equivalently, Y_n).

Equivalently, OPT_i is the solution that agrees with Y_i (equivalently, with X_i) on the first i elements, and with OPT in the remaining ones. In particular, $OPT_0 = OPT$ and $OPT_n = X_n = Y_n$. To bound how much $f(OPT_i)$ decreases as i goes from 0 to n, we can bound its *local changes*, i.e., its changes between two consecutive iterations of the algorithm. This is achieved by the next lemma, that connects this local change to the local changes of the value of the solutions constructed by the algorithm.

Lemma 14. *Let $i \in [n]$. Then $f(OPT_{i-1}) - f(OPT_i) \leq (f(X_i) - f(X_{i-1})) + (f(Y_i) - f(Y_{i-1}))$.*

Proof. Assume wlog that $a \geq b$ at iteration i, the other case following analogously. Then $X_i = X_{i-1} \cup \{e_i\}$ and $Y_i = Y_{i-1}$. Hence,

$$(f(X_i) - f(X_{i-1})) + (f(Y_i) - f(Y_{i-1})) = a.$$

First assume $e_i \in OPT$. Then $OPT_{i-1} = OPT_i$, hence $f(OPT_{i-1}) - f(OPT_i) = 0$. The statement follows since

$$a \geq \frac{1}{2}(a + b) \geq 0,$$

where last inequality follows by Exercise 12.

Hence, we can assume that $e_i \notin OPT$, which implies $OPT_i \setminus OPT_{i-1} = \{e_i\}$. Then:

$$\begin{aligned} f(OPT_{i-1}) - f(OPT_i) &= f(OPT_{i-1}) - f(OPT_{i-1} \cup \{e_i\}) \\ &\leq f(Y_{i-1} \setminus \{e_i\}) - f(Y_i) && \text{(by Lemma 1, since } Y_i \supseteq OPT_{i-1}) \\ &= b && \text{(by definition)} \\ &\leq a && \text{(by hypothesis),} \end{aligned}$$

concluding the proof. □

By summing over all local changes, we have:

$$\underbrace{\sum_{i=1}^{n} f(OPT_{i-1}) - f(OPT_i)}_{f(OPT_0) - f(OPT_n)} \leq \underbrace{\sum_{i=1}^{n} (f(X_i) - f(X_{i-1}))}_{f(X_n) - f(X_0)} + \underbrace{\sum_{i=1}^{n} (f(Y_i) - f(Y_{i-1}))}_{f(Y_n) - f(Y_0)}$$

As $f(Y_0), f(X_0) \geq 0$, $OPT_0 = OPT$ and $X_n = Y_n = OPT_n$ is the output of Algorithm 5, we deduce:

$$f(\underbrace{OPT_0}_{OPT}) - f(\underbrace{OPT_n}_{X_n}) \leq f(X_n) - \underbrace{f(X_0)}_{\geq 0} + f(\underbrace{Y_n}_{X_n}) - \underbrace{f(Y_0)}_{\geq 0} \Rightarrow f(OPT) \leq 3f(X_n),$$

concluding the proof. □

Consider the modification of Algorithm 5 that replaces Steps from 4 to 7 with the following:

> Let $a' = \max(a, 0)$ and $b' = \max(b', 0)$. If $b' = 0$, set $X_i = X_{i-1} \cup \{e_i\}$, $Y_i = Y_{i-1}$. Else, with probability $a'/(a' + b')$, set $X_i = X_{i-1} \cup \{e_i\}$, $Y_i = Y_{i-1}$. With the complement probability, set $X_i = X_{i-1}$, $Y_i = Y_{i-1} \setminus \{e_i\}$.

This algorithm is called *randomized double greedy*.

Theorem 15. *The randomized double greedy outputs a $\frac{1}{2}$-approximation in expectation to* `MAX-C`.

We refer to [2] for the proof of Theorem 15. It can be shown it is NP-Hard to obtain an approximation strictly better than 1/2 for `MAX-C`.

3 Minimizing a submodular function

Unlike its maximization counterpart, the problem `MIN-SF`:

Given: A submodular function $f : 2^V \to \mathbb{R}$.
Find: A set $\emptyset \subseteq X \subseteq V$ that minimizes $f(X)$.

can be solved with a polynomial number of oracle calls. As next example shows, it generalizes the matroid intersection problem.

Example 9. (Matroid intersection via submodular functions) Let $M_1 = (S, I_1)$ and $M_2 = (S, I_2)$ be two matroids on the same finite ground set S, with rank functions r_1 and r_2, respectively. For $A \subseteq S$, let

$$f(A) := r_1(A) + r_2(S \setminus A).$$

Since the rank function of a matroid is submodular, both r_1 and r_2 are submodular. Using Lemma 2 and Exercise 2, we deduce that f is submodular. From the matroid intersection theorem, we know that the maximum size of a common independent set is given by

$$\min f(A) : A \subseteq S.$$

Hence, by minimizing a submodular function, we can obtain the maximum cardinality of a common independent set.

3.1 An algorithm for f symmetric

Assume first that f is symmetric. We call `MIN-SSF` the following problem:

Given: A symmetric submodular function $f : 2^V \to \mathbb{R}$.
Find: A set $\emptyset \subsetneq X \subsetneq V$ that minimizes $f(X)$.

Note that in this case we are excluding $\emptyset$ and its symmetric V from being possible outputs. This is because, for a symmetric submodular function, they are always the minimizers. Indeed, when f is symmetric, for each set $A \subseteq V$, we have:

$$2f(V) = 2f(\emptyset) = f(\emptyset) + f(V) \leq f(A) + f(V \setminus A) = 2f(A).$$

Recall that we have already seen in class an algorithm that solves a special case of `MIN-SSF`, namely, when f is the cut function of a graph. The algorithm we propose is in fact a generalization of that. It has $|V|$ rounds. At the first round, it finds a *special* pair of elements (x, y), with $x \neq y \in V$, with the property that

$$f(\{x\}) = \min\{f(S) \,:\, \{x\} \subseteq S \subseteq V \setminus \{y\}\}. \tag{11}$$

(Note that even the *existence* of a special pair is non-trivial and requires a proof). Hence, x would be the optimal solution of our problem if we knew that exactly one of x and y is in the optimal solution. This may not be the case, however. Hence, we store the set $\{x\}$, and produce a new function f' over subsets of $V' := V \setminus \{x\}$ by "gluing" the nodes x and y as follows:

$$f'(S) = \begin{cases} f(S) & \text{if } y \notin S \\ f(S \cup \{x\}) & \text{otherwise} \end{cases}. \tag{12}$$

It is easy to see that, if f is a symmetric submodular fuction, so is f'. Moreover, the following holds.

Lemma 16. *Let f, x, y, V, V' be defined as above. Let S^* be the optimal solution of `MIN-SSF` on input f', and assume wlog that $y \notin S^*$. Then either $\{x\}$ or S^* is an optimal solution to `MIN-SSF` on input f.*

Proof. Suppose $\{x\}$ is not an optimal solution to `MIN-SSF` on input f. Then, by (11), there is an optimal solution that does not contain either x or y. Call this solution S'. Hence $f'(S') = f(S')$. On the other hand, $f(S^*) = f'(S^*) \leq f'(S') = f(S')$, hence S^* is also an optimal solution. □

The algorithm therefore iteratively finds a special pair, stores the element x found, and creates f', until the ground set is small enough.

The correctness of Algorithm 6 follows from the discussion above. Its running time depends on the time needed to find a special pair. We now show that a special pair can be found by Algorithm 7, hence concluding the following.

Theorem 17. *Algorithm 6 solves `MIN-SSF` with $O(|V|^3)$ oracle calls.*

The correctness of Algorithm 7 follows from the following lemma, where we assume $S_0 = \emptyset$.

Lemma 18. *For $i = 1, \ldots, n-1$, for each $T \subseteq S_{i-1}$ and for each $v \in V \setminus S_i$, the following holds:*

$$f(S_i) + f(\{v\}) \leq f(S_i \setminus T) + f(T \cup \{v\}). \tag{13}$$

Algorithm 6 Algorithm to minimize a symmetric submodular function

Require: A symmetric submodular function $f : 2^V \to \mathbb{R}$.

if $V = \{x, y\}$ **then**
  return $\{x\}$.
end if
Find a special pair (x, y) for f.
create f' from f as in (12).
Recursively call the algorithm on input f', and let S^* be the optimal solution of `MIN-SSF` on input f' that does not contain y.
return $\arg\min\{f(S^*), f(\{x\})\}$.

Algorithm 7 Algorithm that finds a special pair (x, y)

Require: A symmetric submodular function $f : 2^V \to \mathbb{R}$, with $n = |V|$.

Set $S_1 = \{v\}$ for some $v \in V$.
for $i = 2, \ldots, n$ **do**
  for $v \in V \setminus S_{i-1}$ **do**
    Let $key(v) := f(S_{i-1} \cup \{v\}) - f(\{v\})$.
  end for
  Let $\tau_i \in \arg\min\{key(v) : v \in V \setminus S_{i-1}\}$.
  Set $S_i = S_{i-1} \cup \{\tau_i\}$.
end for
return (τ_n, τ_{n-1}).

Proof. We prove the statement by induction on i. When $i = 1$, the only choice is $T = \emptyset$, and (13) reads

$$f(S_1) + f(\{v\}) \leq f(S_1) + f(\{v\})$$

which is clearly true for each $v \in V \setminus S_1$.

Now assume the statement is true up to $i-1$ and we will show the claim for i. Choose $v \in V \setminus S_i$ and $T \subseteq S_{i-1}$, and define $j := \max\{\ell : \tau_\ell \in T\}$. Hence $T \subseteq S_j$. We will consider the following two cases. First, if $j = i - 1$, then

$$\begin{aligned}
f(S_i \setminus T) + f(T \cup \{v\}) &= f(S_{i-1} \cup \{\tau_i\} \setminus T) + f(T \cup \{v\}) \\
&\geq f(S_{i-1}) + f(\{\tau_i\}) - f\big(S_{i-1} \setminus (S_{i-1} \setminus T)\big) + f(T \cup \{v\}) \\
&= f(S_{i-1}) + f(\{\tau_i\}) - f(T) + f(T \cup \{v\}) \\
&\geq f(S_{i-1} \cup \{v\}) + f(\{\tau_i\}) \\
&\geq f(\{v\}) + f(S_{i-1} \cup \{\tau_i\}) \\
&= f(\{v\}) + f(S_i),
\end{aligned}$$

as required. The first inequality is by induction with $i = i - 1$, $T = S_{i-1} \setminus T$ and $v = \tau_i$. The second inequality follows from (1) with $A = S_{i-1}$ and $B = T \cup \{v\}$. The last inequality is by the choice of τ_i.

Exercise 13. Show that, if $j < i - 1$, $f(S_i \setminus T) + f(T \cup \{v\}) \geq f(S_i) + f(\{v\})$.

□

We now conclude the proof that Algorithm 7 finds a special pair. Let $x = \tau_n$ and $y = \tau_{n-1}$. Apply Lemma 18 with $i = n-1$. Take any $\{x\} \subseteq T' \subseteq V \setminus \{y\}$ and set $T := S_{n-2} \setminus T'$. We deduce

$$f(V \setminus \{x\}) + f(\{x\}) \leq f(T') + f(V \setminus T'),$$

and the statement follows from symmetry of f.

3.2 The Lovasz extension and an algorithm for `MIN-SF`

In this section, we see an algorithm for solving `MIN-SF`. Although more efficient (also, more complex) algorithms for `MIN-SF` exist, the one we present in here builds on three nice ingredients:

(a) the Ellipsoid method for convex optimization;

(b) a continuous, convex extension f^- of a discrete function f with domain $\{0,1\}^n$;

(c) an efficiently computable extension f^L of f as above, that coincides with f^- if and only if f is submodular. f^L is called the *Lovasz extension*.

(a) The Ellipsoid method for convex optimization. The Ellipsoid method can be employed to obtain, in polynomial time, the optimum solution to a linear program. Under similar hypothesis, it can be employed to *approximately* solve convex optimization problems in polynomial time. The next statements are not too formal, but will be enough for our purposes. Let $f^- : \mathbb{R}^n \to \mathbb{R}$ be a convex function and $K \subseteq \mathbb{R}^n$ a convex set. Suppose that we have access to an evaluation oracle for both f^- and its gradient: that is, given $\bar{x} \in K$, the oracle outputs $f^-(\bar{x})$ and $\nabla f^-(\bar{x})$. Suppose moreover that we have a separation oracle that, given a point $\bar{x} \in \mathbb{R}^n$, either concludes that $\bar{x} \in K$, or gives an inequality $c^T x \leq \delta$ valid for K but violated by $\bar{x}$. Last, suppose that we know of a "not too large" ellipsoid containing K. Then, we can find a point that is "almost" in K and "almost" realizes the minimum of f^- over K, with a number of calls to the oracles that is polynomial in the size of the problem and in the required approximation guarantee. Under additional hypothesis, some of the conditions above can be dropped and we can actually find the minimum of f^- over K. Those additional hypothesis will be satisfied by our problem. In particular, our goal will be to reduce `MIN-SF` over f to a minimization of a convex function f^- over a convex set K, such that there is a polynomial-time oracle for evaluating f^- and for separating over K. As usual, we assume that to we have a polynomial-time oracle for computing the submodular function f.

(b) The convex closure. Given a function $f : 2^V \to \mathbb{R}$, we let $n = |V|$ and interpret f as a function with domain $\{0,1\}^n$, where we let $x_S \in \{0,1\}^n$ be the point with support S, and we let $f(x_S) = f(S)$ for all $S \subseteq V$. The *convex closure* of f is the function with domain $[0,1]^n$ defined as follows:

$$\begin{aligned} f^-(x) = \quad \min \quad & \{\sum_{S \subseteq V} \alpha_S f(S) : \\ & \sum_{S \subseteq V} \alpha_S x_S = x,\ \sum_{S \subseteq V} \alpha_S = 1,\ \alpha_S \geq 0\}. \end{aligned} \tag{14}$$

For a given $x \in \mathbb{R}^n$, we call a vector that satisfies constraints from (14) a *feasible multiplier* for x; if it moreover achieves the minimum above, we call it an *optimal multiplier* for x.

In order to parse this definition, let's first observe that a "natural" extension of f would associate to a point $x \in [0,1]^n$ that can be obtained as a convex combination of points of the hypercube, the combination, with the same multipliers, of the function evaluated at those points. Note that each feasible multiplier is associated to a convex combinations of points in $\{0,1\}^n$ producing x. However, there are many such combinations, so which one should we pick to define f^-? We choose the one giving the minimum possible value. This explains the notation f^-. Its name is motivated by the following.

Lemma 19. *Let f, f^- be as above. Then f^- is convex.*

Proof. Let $x, y \in [0,1]^n$, $\lambda \in [0,1]$, and $z = \lambda x + (1-\lambda)y$. Let α^x (resp. α^y) be an optimal multiplier for x (resp. y). For $S \subseteq V$, define

$$\alpha^z_S = \lambda \alpha^x_S + (1-\lambda)\alpha^y_S \geq 0$$

Since

$$\sum_{S \subseteq V} \alpha^z_S = \lambda \sum_{S \subseteq V} \alpha^x_S + (1-\lambda) \sum_{S \subseteq V} \alpha^y_S = 1$$

and

$$\sum_{S \subseteq V} \alpha^z_S x_S = \lambda \sum_{S \subseteq V} \alpha^x_S x_S + (1-\lambda) \sum_{S \subseteq V} y_S \alpha^y_S = \lambda x + (1-\lambda) y = z,$$

we deduce that α^z is a feasible multiplier for z. We have therefore

$$f^-(z) \leq \sum_{S \subseteq V} (\lambda \alpha^x_S + (1-\lambda)\alpha^y_S) f(S) = \lambda f(x) + (1-\lambda) f(y),$$

as required. □

Note that the minimum of f^- over $[0,1]^n$ is achieved at a vertex. Therefore, we can solve `MIN-SF` by computing the minimum of f^- over $[0,1]^n$. As clearly $[0,1]^n$ has a polytime separation oracle, all we have left to describe is a polytime oracle for evaluating f^-. This is described next.

(c) The Lovasz extension. Define the *Lovasz extension* $f^L : [0,1]^n \to \mathbb{R}$ of f as follows. For each $x \in [0,1]^n$, we have:

$$f^L(x) := \sum_{i=0}^{n} \lambda^*_i f(S^*_i),$$

where $\emptyset = S^*_0 \subsetneq S^*_1 \subsetneq S^*_2 \subsetneq \cdots \subsetneq S^*_n = V$, and $\lambda^*_0, \lambda^*_1, \ldots, \lambda^*_n$ are defined as follows: suppose entries in $V = \{1, \ldots, n\}$ are ordered so that $x_i \geq x_{i+1}$ for $i \in [n-1]$. Then, set for $i \in [n]$ $S^*_i := \{x_1, \ldots, x_i\}$ and $\lambda^*_i := x_i - x_{i+1}$ (with $x_{n+1} := 0$), and $\lambda^*_0 := 1 - \sum_{i=1}^n \lambda^*_i$.

Observe that λ^* is a feasible multiplier for x (letting $\lambda^*_S = 0$ for all $S^* \neq S^*_0, \ldots, S^*_n$). Indeed, clearly $\lambda^* \geq 0$, $\sum_{i=0}^n \lambda^*_i = 1$, and for $j \in [n]$ we have:

$$(\sum_{i=0}^{n} \lambda^*_i x_{S_i})_j = \sum_{\ell=j}^{n} \lambda^*_\ell = \sum_{\ell=j}^{n} (x_\ell - x_{\ell+1}) = x_j - x_{n+1} = x_j.$$

Hence, $f^L(x) \geq f^-(x)$ for all $x \in [0,1]^n$. Moreover, $f^L(x)$ can be computed efficiently, but we expect in general $f^L \neq f^-$. However, they coincide if and only if f is submodular. Once this is shown, we can conclude that `MIN-SF` can be solved in polynomial time.

We first observe first the following. A family of sets $\emptyset = S_0 \subsetneq S_1 \subsetneq S_1 \subsetneq \ldots S_n = V$ is called a *chain.*

Lemma 20. *Let $x \in [0,1]^n$ and λ be a feasible multiplier for x whose support is a chain $\emptyset = S_0 \subsetneq \cdots \subsetneq S_n = V$. Then $f^L(\bar{x}) = \sum_{i=0}^n \lambda_i f(S_i)$.*

Proof. Observe that, if $i \in S_\ell, j \notin S_\ell$, for some $i, j, \ell \in [n]$, then we have $x_i \geq x_j$. If moreover $x_i = x_j$, then it must be that $\lambda_\ell = 0$. Hence, without loss of generality, we have that $S^*_\ell = S_\ell$ for each $\ell \in [n]$. Then λ satisfies

$$\begin{aligned} \lambda(S^*_1) &= x_1 - x_2 \\ \lambda(S^*_2) &= x_2 - x_3 \\ &\cdots \\ \lambda(S^*_n) &= x_n. \end{aligned}$$

As the constraint matrix of the system is invertible and λ^* is one feasible solution, we conclude that $\lambda = \lambda^*$. □

Lemma 21. *$f^- = f^L$ if and only if f is submodular.*

Proof. Suppose first f is submodular, and let $x \in [0,1]^n$. Among all optimal multipliers α for x, let α^* be one that maximizes $\sum_{S \subseteq V} \alpha_S |S|^2$.

We claim that the support of α is a chain for x, and then the statement follows by Lemma 20. Suppose by contradiction that there exists $A, B \in V$ with $\alpha^*_A \geq \alpha^*_B > 0$, $A \cap B, A \cup B \neq A, B$. Consider $\bar{\alpha}$ constructed from α^* as follows: for $S \subseteq V$, we have

$$\bar{\alpha}_S := \begin{cases} \alpha^*_S - \alpha^*_B & \text{if } S = A, B \\ \alpha^*_S + \alpha^*_B & \text{if } S = A \cup B, A \cap B \\ \alpha^*_S & \text{otherwise.} \end{cases}.$$

Let us first verify that $\bar{\alpha}$ is a feasible multiplier for x. Clearly $\bar{\alpha} \geq 0$ and $\sum_{S \subseteq V} \bar{\alpha}_S = 1$. We have:

$$\sum_{S \subseteq V} \bar{\alpha}_S x_S = \sum_{S \subseteq V, S \neq A, B, A \cap B, A \cup B} \alpha^*_S x_S + \sum_{S = A, B, A \cap B, A \cup B} \bar{\alpha}_S x_S,$$

hence, to show $\sum_{S \subseteq V} \bar{\alpha}_S x_S = x$, it suffices to observe that for $j \in [n]$, we have

$$\chi(j \in A) + \chi(j \in B) = \chi(j \in A \cup B) + \chi(j \in A \cap B),$$

where $\chi(X) = 1$ if event X holds and $\chi(X) = 0$ otherwise. This shows that $\bar{\alpha}$ is a feasible multiplier for x. Moreover, we have:

$$\begin{aligned} \sum_{S \subseteq V} \bar{\alpha}_S f(x_S) &= \sum_{S \subseteq V, S \neq A, B, A \cap B, A \cup B} \alpha^*_S f(x_S) + \sum_{S = A, B, A \cap B, A \cup B} \bar{\alpha}_S f(x_S) \\ &= \sum_{S \subseteq V} \alpha^*_S f(x_S) + \alpha^*_B (f(A \cup B) + f(A \cap B) - (f(A) + f(B))) \\ &\leq \sum_{S \subseteq V} \alpha^*_S f(x_S), \end{aligned}$$

where last inequality holds by submodularity.

Hence, $\bar{\alpha}$ satisfies $f^-(x) = \sum_{S \subseteq V} \bar{\alpha}_S f(S)$. On the other hand,

$$\sum_{S \subseteq V} \bar{\alpha}_S |S|^2 = \sum_{S \subseteq V} \alpha^*_S |S|^2 + \alpha^*_B (|A \cup B|^2 + |A \cap B|^2 - (|A|^2 + |B|^2)) > \sum_{S \subseteq V} \alpha^*_S |S|^2,$$

a contradiction, where in the last inequality we used that

$$|A\cup B|^2+|A\cap B|^2=(|A|+|B\setminus A|)^2+(|B|-|B\setminus A|)^2=|A|^2+|B|^2+2|B\setminus A|(|A|-|B|+|B\setminus A|).$$

We now show the other direction. Assume that f is not submodular. Then, by Exercise 1, there exists $i,j\notin S\subseteq V$ such that

$$f(S\cup\{i\})-f(S)<f(S\cup\{i,j\})-f(S\cup\{j\}). \tag{15}$$

Let $x=x_S+\frac{1}{2}x_{\{i,j\}}$. We have $\lambda^*_{S\cup\{i,j\}}=\frac{1}{2}$, $\lambda^*_S=\frac{1}{2}$, hence $f^L(x)=\frac{1}{2}f(S)+\frac{1}{2}f(S\cup\{i,j\})$. On the other hand a feasible multiplier α is:

$$\alpha_T=\begin{cases}\frac{1}{2} & \text{if } T=S\cup\{i\} \text{ or } T=S\cup\{j\}\\ 0 & \text{otherwise,}\end{cases}$$

and using (15), we have:

$$f^-(x)\leq\frac{1}{2}f(S\cup\{i\})+\frac{1}{2}f(S\cup\{j\})<\frac{1}{2}f(S)+\frac{1}{2}f(S\cup\{i,j\})=f^L(x),$$

showing $f^L\neq f^-$. □

Exercise 14. In this exercise, we see another proof that the Lovasz extension of a submodular function is convex. Let us start by proving an auxiliary fact.

(a) Let $K,K'\subseteq\mathbb{R}^n$ be bounded convex sets, and for $x\in K'$, let $g(x):=\max_{z\in K}x^Tz$. Prove that $g(x)$ is convex over K'.

We can assume without loss of generality $f(\emptyset)=0$, since we can always translate f so that the equality holds. For questions (b)-(c)-(d) below, fix $x\in[0,1]^n$. Define $z^*\in\mathbb{R}^n$ as follows: for $i\in V$, let $z^*_i=f(S^*_i)-f(S^*_{i-1})$.

(b) Show $x^Tz^*=f^L(x)$.

Now consider the following LP (recall here that x is fixed, and z is the set of variables):

$$(P)\quad\begin{array}{lllll}\max & x^Tz & & & \\ \text{s.t.} & z(S) & \leq & f(S) & \text{for } S\subseteq V\\ & z(V) & = & f(V) & \end{array}$$

(c) Show that z^* is feasible for (P).

(d) Using LP duality, show that z^* is optimal for (P).

(e) Conclude that f^L is convex.

3.2.1 An application of the Lovasz's extension: the fractional Sandwich theorem

For a finite set V, a function $f : 2^V \to \mathbb{R}$ is *supermodular* if $-f$ is submodular or, equivalently, if for every $A, B \subseteq 2^V$ we have:

$$f(A) + f(B) \leq f(A \cap B) + f(A \cup B). \tag{16}$$

A function that is submodular and supermodular is called *modular*. As next exercise show, modular functions can be thought of as the discrete analogous of linear functions.

Exercise 15. Let $f : 2^V \to \mathbb{R}$. f is modular if and only if there exist $w : V \to \mathbb{R}$ such that $f(S) = \sum_{e \in S} w(e)$ for every $S \subseteq V$.

The following is a discrete analogous of a famous result on convex functions.

Theorem 22 (Sandwich Theorem)**.** *Let $f, g : 2^V \to \mathbb{R}$, with f submodular, g supermodular and $f(S) \geq g(S)$ for all $S \subseteq V$ (in short, we write $f \geq g$). Then there exists a modular function $h : 2^V \to \mathbb{R}$ with $f \geq h \geq g$. Moreover, if f, g are integer-valued, so is h.*

The first statement of the Sandwich Theorem easily follows from the results from the previous section (the second is substantially harder to prove). For f, g as in the statement of the theorem, f^- in convex and g^- is concave. By construction, $f^- \geq g^-$. Hence, we can apply a classical result from convexity theory and conclude that there exists a linear function $h^* : [0,1]^V \to \mathbb{R}$ such that $g^- \leq h^* \leq f^-$. Using Exercise 15, we conclude that the restriction of h^* to $\{0,1\}^V$ – call if h – is modular. By construction, $g \leq h \leq f$.

4 Solutions to selected exercises

Solution to Exercise 2. For $A, B \subseteq V$, we have:

$$\begin{aligned} g(A) + g(B) &= f(V \setminus A) + f(V \setminus B) \\ &\geq f((V \setminus A) \cap (V \setminus B)) + f((V \setminus A) \cup (V \setminus B)) \\ &= f(V \setminus (A \cup B)) + f(V \setminus (A \cap B)) \\ &= g(A \cup B) + g(A \cap B). \end{aligned}$$

Solution to Exercise 3. For disjoint sets $U, U' \subseteq V$, denote by $\delta(U, U')$ the set of all edges with one endpoint in U, and the other in U'. Let $A, B \subseteq V$. We have:

$$\begin{aligned} &\delta(A) = \delta(A \cap B, B \setminus A) + \delta(A \setminus B, B \setminus A) + \delta(A \setminus B, V \setminus (A \cup B)) + \delta(A \cap B, V \setminus (A \cup B)); \\ &\delta(B) = \delta(A \cap B, A \setminus B) + \delta(A \setminus B, B \setminus A) + \delta(B \setminus A, V \setminus (A \cup B)) + \delta(A \cap B, V \setminus (A \cup B)); \\ &\delta(A \cup B) = \delta(A \cap B, A \setminus B) + \delta(A \setminus B, V \setminus (A \cup B)) + \delta(B \setminus A, V \setminus (A \cup B)); \\ &\delta(A \cap B) = \delta(A \cap B, A \setminus B) + \delta(A \cap B, B \setminus A) + \delta(A \cap B, V \setminus (A \cup B)). \end{aligned}$$

We conclude therefore:

$$w(\delta(A)) + w(\delta(B)) = w(\delta(A \cup B)) + w(\delta(A \cap B)) + 2w(\delta(A \setminus B, B \setminus A)) \geq w(\delta(A \cup B)) + w(\delta(A \cap B)),$$

as required, where in the last inequality we used $w \geq 0$.

Solution to Exercise 4. Let $A, B \subseteq E$. Take any vertex $v \in V$ and consider the following three cases. If v is adjacent to an edge that is in exactly one of A and B, then it contributes 1 to both $f(A) + f(B)$ and $f(A \cap B) + f(A \cup B)$. If v is not adjacent to any edge of $A \cup B$, then it does not contribute to any of $f(A), f(B), f(A \cap B), f(A \cup B)$. Lastly, if v is adjacent to both A and B, it contributes 2 to $f(A) + f(B)$. Thus, we conclude $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$, as required.

Solution to Exercise 5. See [1].

Solution to Exercise 6. Let $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{Z}$, and s be a string. If s contributes to $I_{CG}(\mathcal{X} \cap \mathcal{Y})$ but not to $I_{CG}(\mathcal{X} \cup \mathcal{Y})$, then s is a substring of $S \in \mathcal{X} \cap \mathcal{Y}$ and of $S' \in (\mathcal{X} \setminus \mathcal{Y}) \cup (\mathcal{Y} \setminus \mathcal{X})$. Hence, it contributes to at least one of $I_{CG}(\mathcal{X})$ and $I_{CG}(\mathcal{Y})$. A similar conclusion holds if s contributes to $I_{CG}(\mathcal{X} \cup \mathcal{Y})$ but not to $I_{CG}(\mathcal{X} \cap \mathcal{Y})$. Now suppose s contributes to both $I_{CG}(\mathcal{X} \cup \mathcal{Y})$ and $I_{CG}(\mathcal{X} \cap \mathcal{Y})$. This means there exist $S \in \mathcal{X} \cap \mathcal{Y}$ and $S' \in \mathcal{Z} \setminus (\mathcal{X} \cup \mathcal{Y})$ that have s as a subsequence. Hence, s contributes to both $I_{CG}(\mathcal{X})$ and $I_{CG}(\mathcal{Y})$. Hence (1) holds.

Solution to Exercise 9. We modify the proof of the greedy algorithm, employing the same notation. We have:

$$\begin{aligned} f(S^*) - f(S_\ell) &\leq f(S^* \cup S_\ell) - f(S_\ell) && \text{(by monotonicity)} \\ &\leq \tfrac{1}{q(f)} \textstyle\sum_{i=1}^{k} (f(S_\ell \cup \{e_i\}) - f(S_\ell)) && \text{(by definition of } q(f)) \\ &\leq \tfrac{k}{q(f)} (f(S_{\ell+1}) - f(S_\ell)) && \text{(by the greedy procedure).} \end{aligned}$$

Hence, we have:

$$f(S_{\ell+1}) - f(S_\ell) \geq \frac{q(f)}{k}(f(S^*) - f(S_\ell)) \;\Leftrightarrow\; f(S^*) - f(S_{\ell+1}) \leq (1 - \frac{q(f)}{k})(f(S^*) - f(S_\ell)).$$

We conclude:

$$\begin{aligned} f(S^*) - f(S_k) &\leq (1 - \tfrac{q(f)}{k})(f(S^*) - f(S_{k-1}) \\ &\leq (1 - \tfrac{q(f)}{k})^2(f(S^*) - f(S_{k-2})) \\ &\leq \ldots \leq (1 - \tfrac{q(f)}{k})^k((f(S^*) - f(S_0)) \\ &\leq (1 - \tfrac{q(f)}{k})^k f(S^*) && \text{(by nonnegativity)} \\ &\leq e^{-q(f)} f(S^*). \end{aligned}$$

Solution to Exercise 10. Repeat the solution of Exercise 9, this time with S^* being the optimal solution to our new problem (note that, in the analysis, we never used the optimality of S^*). Hence, at every step ℓ of the algorithm, we have:

$$f(S^*) - f(S_\ell) \leq (1 - \frac{q(f)}{k^*})^\ell f(S^*),$$

where $k^* := |S^*|$. Hence, if we let $\bar{k} = |\overline{S}|$, we have:

$$f(S^*) - f(S_{\bar{k}-1}) \leq (1 - \frac{q(f)}{k^*})^{\bar{k}-1} f(S^*) \leq e^{-\frac{q(f)}{k^*}(\bar{k}-1)} f(S^*) \Leftrightarrow 1 \leq e^{-\frac{q(f)}{k^*}(\bar{k}-1)} \frac{f(S^*)}{f(S^*) - f(S_{\bar{k}-1})}.$$

Taking the logarithm, we obtain:

$$0 \leq -\frac{q(f)}{k^*}(\bar{k}-1) + \log(\frac{f(S^*)}{f(S^*) - f(S_{\bar{k}-1})}) \leq -\frac{q(f)}{k^*}(\bar{k}-1) + \log(\frac{t}{t - f(S_{\bar{k}-1})}) \leq -\frac{q(f)}{k^*}(\bar{k}-1) + \log t,$$

where in the second inequality we used $f(S^*) \geq t$ and in the third $f(S^*) - f(S_{\bar{k}-1}) \geq 1$, since f is integer-valued. Rearranging gives the thesis.

Solution to Exercise 11. By hypothesis, for $\ell \leq \ell^* - 2$, we have $S_\ell \cup \{e_{\ell+1}\} = S_{\ell+1}$. By iteratively applying the last inequality deduced in the main part of the proof, we conclude

$$\begin{aligned} g(S_{\ell^*-1} \cup \{e_{\ell^*}\}) &\geq g(S^*) + (1 - \tfrac{w(e_{\ell^*})}{B'})(g(S_{\ell^*-1}) - g(S^*)) \\ &\geq g(S^*) + (1 - \tfrac{w(e_{\ell^*})}{B'})(1 - \tfrac{w(e_{\ell^*-1})}{B'})(g(S_{\ell^*-2}) - g(S^*)) \\ &\geq \ldots \geq g(S^*) + \textstyle\prod_{j=1,\ldots,\ell^*}(1 - \frac{w(e_j)}{B'})(g(Y) - g(S^*)) \\ &\geq g(S^*)(1 - (\textstyle\prod_{j=1,\ldots,\ell^*}(1 - \frac{w(e_j)}{B'}))) && \text{(using } g(Y) = 0) \\ &\geq g(S^*)(1 - e^{-(\sum_{j=1}^{\ell^*} w(e_j)/B')}) && \text{(using } (1-x) \leq e^{-x}) \\ &\geq g(S^*)(1 - \tfrac{1}{e}) && \text{(since } \textstyle\sum_{j=1}^{\ell^*} w(e_j) > B'). \end{aligned}$$

Solution to Exercise 12. Observe that, at every iteration i, $(X_i \cup \{e_i\}) \cup (Y_i \setminus \{e_i\}) = Y_{i-1}$ and $(X_i \cup \{e_i\}) \cap (Y_i \setminus \{e_i\}) = X_{i-1}$. Then using submodularity:

$$a + b = f(\underbrace{X_i \cup \{e_i\}}_{A}) - f(\underbrace{X_{i-1}}_{A \cap B}) + f(\underbrace{Y_i \setminus \{e_i\}}_{B}) - f(\underbrace{Y_{i-1}}_{A \cap B}) \geq 0.$$

Solution to Exercise 13. We have:

$$\begin{aligned} f(S_i \setminus T) + f(T \cup \{v\}) &\geq f(S_i \setminus T) + f(S_{j+1}) - f(S_{j+1} \setminus T) + f(\{v\}) \\ &\geq f(S_{j+1} \setminus T) + f(S_i) - f(S_{j+1} \setminus T) + f(\{v\}) \\ &= f(S_i) + f(\{v\}), \end{aligned}$$

as required. The first inequality is by induction with $i = j + 1$, $T = T$, $v = v$, and the second inequality is by (1) with $A = S_i \setminus T$ and $B = S_{j+1}$.

Solution to Exercise 14.

(a) Let $y, x \in K'$, $\lambda \in [0, 1]$, and $w = \lambda x + (1 - \lambda)y$. Then

$$g(w) = \max_{z \in K'} w^T z = \max_{z \in K'} (\lambda(x^T z) + (1-\lambda)(y^T z)) \leq \lambda \max_{z \in K'} (x^T z) + (1-\lambda) \max_{z \in K'} (y^T z) = \lambda g(x) + (1-\lambda) g(y).$$

(b)

$$x^T z^* = \sum_{i=1}^{n} x_i (f(S_i^*) - f(S_{i-1}^*)) = \sum_{i=1}^{n-1} (x_i - x_{i+1}) f(S_i^*) + x_n f(S_n^*) = \sum_{i=1}^{n} \lambda_i^* f(S_i^*) = f^L(x),$$

where in the last equality we used the definition of f^L and that we assumed $f(\emptyset) = 0$.

Before we move on, it is useful to gain some intuition on why this is a reasonable candidate to the optimal solution of (P). By hypothesis, z_1 has the highest coefficient in the objective function. Hence, we would like the corresponding variable to be as large as possible. Since we must have

$$z_1 \leq f(\{z_1\}) = f(S_1^*),$$

we set $z_1^* = f(S_1^*) - f(S_0^*) = f(S_1^*)$. Next in line is z_2. Since it must hold

$$z_1 + z_2 \leq f(\{z_1, z_2\}) = f(S_2^*)$$

and we already set $z_1^* = f(S_1^*)$, we set $z_2^* = f(S_2^*) - z_1^* = f(S_2^*) - f(S_1^*)$, etc. Hence, z^* can be seen as the outcome of a greedy procedure, where at each step we choose to the unassigned variable with highest coefficient the largest possible value.

(c) $f(V) = z^*(V)$ follows by telescopic sum. We show $z^*(S) \leq f(S)$ for all $S \subseteq V$ by induction on $|S|$. For base case $S = \emptyset$, $z^*(S) = f(S) = 0$. Now consider S and take i to be the largest index in S. Take $A = S$ and $B = S_i^* \setminus \{i\}$ and apply the definition of submodularity $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$ with $A = S_i^* \setminus \{i\}$ and $B = S$:

$$f(S) \geq f(S_i^*) - f(\underbrace{S_i^* \setminus \{i\}}_{S_{i-1}^*}) + f(S \setminus \{i\}) = z_i^* + f(S \setminus \{i\}) \geq z_i^* + z^*(S \setminus \{i^*\}) = z^*(S),$$

where the last inequality follows by inductive hypothesis.

(d) The dual of (P) is as follows:

$$\begin{array}{rlllll} & \min & \sum_{S\subseteq V} f(S) y_S & & & \\ (D) & \text{s.t.} & \sum_{S\subseteq V: i\in S} y_S & = & x_i & \text{for } i \in V \\ & & y_S & \geq & 0 & \text{for } S \subsetneq V. \end{array}$$

Observe that, for $i \in \{0, \ldots, n\}$, we have:

$$z^*(S_i^*) = \sum_{\ell=i}^{n} (f(S_i^*) - f(S_{i+1}^*)) = f(S_i^*).$$

Hence, because of complementary slackness, we define an optimal dual solution that is possibly non-zero in sets S_i^* (only). For $S \subseteq V$, we define the following dual variables y^*:

$$y_S^* = \begin{cases} x_n & \text{if } S = V \\ x_i - x_{i+1} & \text{if } S = S_i^* \text{ for some } i \in [n-1] \\ 0 & \text{otherwise.} \end{cases}$$

By construction, $x_i \geq x_{i+1}$ for all $i \in [n-1]$, thus $y^* \geq 0$. On the other hand, for $i \in [n]$, we have:

$$\sum_{S: i\in S} y_S^* = \sum_{\ell=i}^{n-1} (x_\ell - x_{\ell+1}) + x_n = x_i,$$

hence y^* is feasible for (D). Since (y^*, z^*) satisfy the complementary slackness, the thesis follows.

(e) We have shown that the optimum of (P) has value $x^T z^* = f^L(x)$, and the thesis follows immediately from applying part (a).

5 Sources

Properties and examples from Section 1 are mostly classical, with Example 4 appearing in [1], Example 6 appearing in [7] and concepts from Section 1.2 appearing in [3]. Results from Section 2.1 appeared in [9]. Results from Section 2.2 appeared in [11]. Results from Section 2.4 appeared in [2]. Results from Section 2.3 have appeared in [4]. Results from Section 3.1 appeared in [10]. An algorithm for minimizing a submodular function using the ellipsoid method appeared in [6], while our treatment of the topic is heavily inspired by [8; 12]. The Sandwhich Theorem has appeared in [5].

References

[1] Aprile, M., Conforti, M., Faenza, Y., Fiorini, S., Huynh, T., Macchia, M. (2020). Recognizing Cartesian products of matrices and polytopes. Proceedings of CTW 2020.

[2] Buchbinder, N., Feldman, M., Naor, J., and Schwartz, R. (2015). A tight linear time (1/2)-approximation for unconstrained submodular maximization. SIAM Journal on Computing, 44(5), 1384–1402.

[3] Das, A. and Kempe, D. (2018). Approximate submodularity and its applications: Subset selection, sparse approximation and dictionary selection. The Journal of Machine Learning Research, 19(1), 74–107.

[4] Fisher, M. L., Nemhauser, G. L., and Wolsey, L. A. (1978). An analysis of approximations for maximizing submodular set functions—II. In Polyhedral combinatorics (pp. 73–87). Springer, Berlin, Heidelberg.

[5] Frank, A. Finding feasible vectors of Edmonds-Giles polyhedra. Journal of Combinatorial Theory, Series B 36.3 (1984): 221–239.

[6] Grötschel, M., Lovász, L., and Schrijver, A. (1984). Geometric algorithms and combinatorial optimization.

[7] D. Kempe, J. Kleinberg, and E. Tardos (2003). Maximizing the spread of influence through a social network. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 137–146.

[8] Kothari, P. (2016) Submodular Functions, Lovasz Extension and Minimization. Lecture from the class COS521: Advanced Algorithm Design, Princeton.

[9] Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions—I. Mathematical programming, 14(1), 265–294.

[10] Queyranne, M. (1998). Minimizing symmetric submodular functions. Mathematical Programming, 82(1-2), 3-12.

[11] Sviridenko, M. (2004). A note on maximizing a submodular set function subject to a knapsack constraint. Operations Research Letters 32.1: 41–43.

[12] Vondràk, J. (2010). Continuous extensions of submodular functions. Lecture from the class CS369P: Polyhedral techniques in combinatorial optimization, Stanford.