2
Greedy Strategy

Someone reminded me that I once said, "Greed is good."
Now it seems that it's legal.

— Gordon Gekko (in *Wall Street: Money Never Sleeps*)

I think greed is healthy. You can be greedy
and still feel good about yourself.

— Ivan Boesky

The greedy strategy is a simple and popular idea in the design of approximation algorithms. In this chapter, we study two general theories, based on the notions of independent systems and submodular potential functions, about the analysis of greedy algorithms, and present a number of applications of these methods.

2.1 Independent Systems

The basic idea of a greedy algorithm can be summarized as follows:

(1) We define an appropriate *potential function* $f(A)$ on potential solution sets A.

(2) Starting with $A = \emptyset$, we grow the solution set A by adding to it, at each stage, an element that maximizes (or, minimizes) the value of $f(A \cup \{x\})$, until $f(A)$ reaches the maximum (or, respectively, minimum) value.

D.-Z. Du et al., *Design and Analysis of Approximation Algorithms*,
Springer Optimization and Its Applications 62, DOI 10.1007/978-1-4614-1701-9_2,

We first consider a simple setting, in which the potential function is the same as the objective function. In the following, we write $\mathbb{N}^+$ to denote the set of positive integers, and $\mathbb{R}^+$ the set of nonnegative real numbers.

Let E be a finite set and $\mathcal{I}$ a family of subsets of E. The pair $(E, \mathcal{I})$ is called an *independent system* if

(I_1) $I \in \mathcal{I}$ and $I' \subseteq I \Rightarrow I' \in \mathcal{I}$.

Each subset in $\mathcal{I}$ is called an *independent subset*. Let $c : E \to \mathbb{R}^+$ be a nonnegative function. For every subset F of E, define $c(F) = \sum_{e \in F} c(e)$. Consider the following problem:

> MAXIMUM INDEPENDENT SUBSET (MAX-ISS): Given an independent system $(E, \mathcal{I})$ and a cost function $c : E \to \mathbb{R}^+$,
>
> $$\begin{array}{ll} \text{maximize} & c(I) \\ \text{subject to} & I \in \mathcal{I}. \end{array}$$

We remark that the family $\mathcal{I}$ has, in general, an exponential size and cannot be given explicitly (and, hence, an exhaustive search for the maximum $c(I)$ is impractical). In most applications, however, the system $(E, \mathcal{I})$ is given in such a way that the condition of whether $I \in \mathcal{I}$ can be determined in polynomial time. Under this assumption, the following greedy algorithm, which uses the objective function c as the potential function, works in polynomial time.

Algorithm 2.A (*Greedy Algorithm for* MAX-ISS)

Input: An independent system $(E, \mathcal{I})$ and a cost function $c : E \to \mathbb{R}^+$.

(1) Sort all elements in $E = \{e_1, e_2, \ldots, e_n\}$ in the decreasing order of c. Without loss of generality, assume that $c(e_1) \geq c(e_2) \geq \cdots \geq c(e_n)$.

(2) Set $I \leftarrow \emptyset$.

(3) **For** $i \leftarrow 1$ **to** n **do**
    if $I \cup \{e_i\} \in \mathcal{I}$ **then** $I \leftarrow I \cup \{e_i\}$.

(4) Output $I_G \leftarrow I$. ■

For any instance $(E, \mathcal{I}, c)$ of the problem MAX-ISS, let I^* be its optimal solution and I_G the independent set produced by Algorithm 2.A. We will see that $c(I_G)/c(I^*)$ has a simple upper bound that is independent of the cost function c.

For any $F \subseteq E$, a set $I \subseteq F$ is called a *maximal independent subset* of F if no independent subset of F contains I as a proper subset. For any set $I \subseteq E$, let $|I|$ denote the number of elements in I. Define

$$\begin{aligned} u(F) &= \min\{|I| \mid I \text{ is a maximal independent subset of } F\}, \\ v(F) &= \max\{|I| \mid I \text{ is an independent subset of } F\}. \end{aligned} \tag{2.1}$$

Theorem 2.1 *The following inequality holds for any independent system* $(E, \mathcal{I})$ *and any function* $c : E \to \mathbb{R}^+$*:*

$$1 \leq \frac{c(I^*)}{c(I_G)} \leq \max_{F \subseteq E} \frac{v(F)}{u(F)}.$$

Proof. Assume that $E = \{e_1, e_2, \ldots, e_n\}$, and $c(e_1) \geq \cdots \geq c(e_n)$. Denote $E_i = \{e_1, \ldots, e_i\}$. We claim that $E_i \cap I_G$ is a maximal independent subset of E_i. To see this, we assume, by way of contradiction, that this is not the case; that is, there exists an element $e_j \in E_i \setminus I_G$ such that $(E_i \cap I_G) \cup \{e_j\}$ is independent. Now, consider the jth iteration of the loop of step (3) of Algorithm 2.A. The set I at the beginning of the jth iteration is a subset of I_G, and so $I \cup \{e_j\}$ must be a subset of $(E_i \cap I_G) \cup \{e_j\}$ and, hence, is an independent set. Therefore, the algorithm should have added e_j to I in the jth iteration. This contradicts the assumption that $e_j \notin I_G$.

From the above claim, we see that

$$|E_i \cap I_G| \geq u(E_i).$$

Moreover, since $E_i \cap I^*$ is independent, we have

$$|E_i \cap I^*| \leq v(E_i).$$

Now, we express $c(I_G)$ and $c(I^*)$ in terms of $|E_i \cap I_G|$ and $|E_i \cap I^*|$, respectively. We note that for each $i = 1, 2, \ldots, n$,

$$|E_i \cap I_G| - |E_{i-1} \cap I_G| = \begin{cases} 1, & \text{if } e_i \in I_G, \\ 0, & \text{otherwise.} \end{cases}$$

Therefore,

$$\begin{aligned} c(I_G) &= \sum_{e_i \in I_G} c(e_i) = c(e_1) \cdot |E_1 \cap I_G| + \sum_{i=2}^{n} c(e_i) \cdot (|E_i \cap I_G| - |E_{i-1} \cap I_G|) \\ &= \sum_{i=1}^{n-1} |E_i \cap I_G| \cdot (c(e_i) - c(e_{i+1})) + |E_n \cap I_G| \cdot c(e_n). \end{aligned}$$

Similarly,

$$c(I^*) = \sum_{i=1}^{n-1} |E_i \cap I^*| \cdot (c(e_i) - c(e_{i+1})) + |E_n \cap I^*| \cdot c(e_n).$$

Denote $\rho = \max_{F \subseteq E} v(F)/u(F)$. Then we have

$$\begin{aligned} c(I^*) &\leq \sum_{i=1}^{n-1} v(E_i) \cdot (c(e_i) - c(e_{i+1})) + v(E_n) \cdot c(e_n) \\ &\leq \sum_{i=1}^{n-1} \rho \cdot u(E_i) \cdot (c(e_i) - c(e_{i+1})) + \rho \cdot u(E_n) \cdot c(e_n) \leq \rho \cdot c(I_G). \quad \square \end{aligned}$$

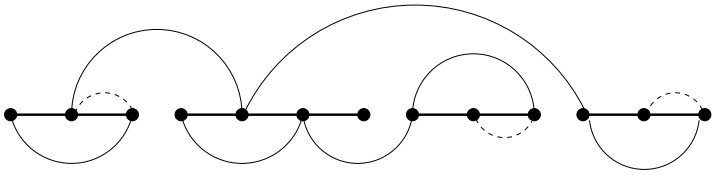


Figure 2.1: Two maximal independent subsets I and J for the problem MAX-HC (the thick lines indicate edges of I, the thin curves and dotted curves indicate the edges of J, and the dotted curves indicate edges shared by I and J).

We note that the ratio $\rho = \max_{F \subseteq E} v(F)/u(F)$ depends only on the structure of the family $\mathcal{I}$ and is independent of the cost function c. Thus, this upper bound is often easy to calculate. We demonstrate the application of this property in two examples.

First, consider the problem MAX-HC defined in Section 1.5. Each instance of this problem consists of n vertices and a distance table on these n vertices. The problem is to find a Hamiltonian circuit of the maximum total distance. Let E be the edge set of the complete graph on the n vertices. Let $\mathcal{I}$ be the family of subsets of E such that $I \in \mathcal{I}$ if and only if I is either a Hamiltonian circuit or a union of disjoint paths (i.e., paths that do not share any common vertex). Clearly, $(E, \mathcal{I})$ is an independent system and whether or not I is in $\mathcal{I}$ can be determined in polynomial time. That is, the problem MAX-HC is a special case of the problem MAX-ISS, and Algorithm 2.A runs on MAX-HC in polynomial time.

Lemma 2.2 *Let $(E, \mathcal{I})$ be the independent system defined above, and F a subset of E. Suppose that I and J are two maximal independent subsets of F. Then $|J| \leq 2|I|$.*

Proof. For $i = 1, 2$, let V_i denote the set of vertices of degree i in I. That is, V_1 is the set of end vertices in I and V_2 is the set of intermediate vertices in I. Clearly, $|I| = |V_2| + |V_1|/2$. Since I is a maximal independent subset of F, every edge in F either is incident on a vertex in V_2 or connects two endpoints of a path in I. Let J_2 be the set of edges in J incident on a vertex in V_2, and $J_1 = J \setminus J_2$. Since J is an independent set, at most two edges in J_2 could be incident on each vertex in V_2. That is, $|J_2| \leq 2|V_2|$. Moreover, every edge in J_1 must connect two endpoints in V_1 in a path of I, and at most one edge in J_1 could be incident on each vertex in V_1. Therefore, $|J_1| \leq |V_1|/2$. (Figure 2.1 shows an example of maximal independent subsets I and J.) Together, we have

$$|J| = |J_1| + |J_2| \leq \frac{|V_1|}{2} + 2|V_2| \leq 2|I|. \qquad \square$$

Theorem 2.3 *When it is applied to the problem* MAX-HC, Algorithm 2.A *is a polynomial-time 2-approximation.*

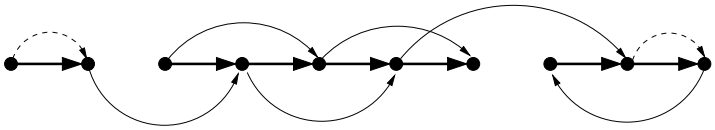


Figure 2.2: Two maximal independent subsets I and J for the problem MAX-DHP.

A similar application gives us a rather weaker performance ratio for the problem MAX-DHP, also defined in Section 1.5. An instance of this problem consists of n vertices and a directed distance table on these n vertices. The problem is to find a directed Hamiltonian path of the maximum total distance. Let E be the set of edges of the complete directed graph on the n vertices. Let $\mathcal{I}$ be the family of subsets of E such that $I \in \mathcal{I}$ if and only if I is a union of disjoint paths. Clearly, $(E, \mathcal{I})$ is an independent system, and whether or not I is in $\mathcal{I}$ can be determined in polynomial time.

Lemma 2.4 *Let $(E, \mathcal{I})$ be the independent system defined as above, and F a subset of E. Suppose that I and J are two maximal independent subsets of F. Then $|J| \leq 3|I|$.*

Proof. Since I is a maximal independent subset of F, every edge in F must have one of the following properties:

(1) It shares a head with an edge in I;

(2) It shares a tail with an edge in I; or

(3) It connects from the head to the tail of a maximal path in I.

(Figure 2.2 shows an example of two maximal independent subsets I and J.)

Let J_1, J_2, and J_3 be the subsets of edges in J that have properties (1), (2) and (3), respectively. Since J is an independent subset, each edge in I can share its head (or its tail) with at most one edge in J, and each maximal path in I can be connected from the head to the tail by at most one edge in J. That is, $|J_i| \leq |I|$, for $i = 1, 2, 3$. Thus,

$$|J| = |J_1| + |J_2| + |J_3| \leq 3|I|. \qquad \square$$

Theorem 2.5 *When it is applied to the problem* MAX-DHP*,* Algorithm 2.A *is a polynomial-time* 3*-approximation.*

The following simple example shows that the performance ratio given by the above theorem cannot be improved.

Example 2.6 Consider the following distance table on four vertices, in which the parameter ε is a positive real number less than 1:

	a	b	c	d
a	0	1	ε	ε
b	ε	0	1	ε
c	ε	$1+\varepsilon$	0	1
d	ε	ε	ε	0

It is clear that the longest Hamiltonian path has distance 3 and yet the greedy algorithm selects the edge (c, b) first and gets a path of total distance $1 + 3\varepsilon$. The performance ratio is, thus, equal to $3/(1+3\varepsilon)$, which approaches 3 when ε approaches zero. □

2.2 Matroids

Let E be a finite set and $\mathcal{I}$ a family of subsets of E. The pair $(E, \mathcal{I})$ is called a *matroid* if

(I_1) $I \in \mathcal{I}$ and $I' \subseteq I \Rightarrow I' \in \mathcal{I}$; and

(I_2) For any subset F of E, $u(F) = v(F)$,

where $u(F)$ and $v(F)$ are the two functions defined in (2.1). Thus, an independent system $(E, \mathcal{I})$ is a matroid if and only if, for any subset F of E, all maximal independent subsets of F have the same cardinality. From Theorem 2.1, we know that Algorithm 2.A produces an optimal solution for the problem MAX-ISS if the input instance $(E, \mathcal{I})$ is a matroid. The next theorem shows that this property actually characterizes the notion of matroids.

Theorem 2.7 *An independent system $(E, \mathcal{I})$ is a matroid if and only if for every nonnegative function $c : E \to \mathbb{R}^+$, the greedy* Algorithm 2.A *produces an optimal solution for the instance $(E, \mathcal{I}, c)$ of* MAX-ISS.

Proof. The "only if" part is just Theorem 2.1. Now, we prove the "if" part. Suppose that $(E, \mathcal{I})$ is not a matroid. Then we can find a subset F of E such that F has two maximal independent subsets I and I' with $|I| > |I'|$. Define, for any $e \in E$,

$$c(e) = \begin{cases} 1+\epsilon, & \text{if } e \in I', \\ 1, & \text{if } e \in I \setminus I', \\ 0, & \text{if } e \in E \setminus (I \cup I'), \end{cases}$$

where ϵ is a positive number less than $1/|I'|$ (so that $c(I) > c(I')$). Clearly, for this cost function c, Algorithm 2.A produces the solution set I', which is not optimal. □

The following are some examples of matroids.

Example 2.8 Let E be a finite set of vectors and $\mathcal{I}$ the family of linearly independent subsets of E. Then the size of the maximal independent subset of a subset $F \subseteq E$ is the rank of F and is unique. Thus, $(E, \mathcal{I})$ is a matroid. □

Example 2.9 Given a graph $G = (V, E)$, let $\mathcal{I}$ be the family of edge sets of acyclic subgraphs of G. Then it is clear that $(E, \mathcal{I})$ is an independent system. We verify that it is actually a matroid, which is usually called a *graph matroid*.

Consider a subset F of E. Suppose that the subgraph (V, F) of G has m connected components. We note that in each connected component C of (V, F), a maximal acyclic subgraph is just a spanning tree of C, in which the number of edges is exactly one less than the number of vertices in C. Thus, every maximal acyclic subgraph of (V, F) has exactly $|V| - m$ edges. So, condition (I_2) holds for the independent system $(E, \mathcal{I})$, and hence $(E, \mathcal{I})$ is a matroid. □

Example 2.10 Consider a directed graph $G = (V, E)$ and a nonnegative integer function f on V. Let $\mathcal{I}$ be the family of edge sets of subgraphs whose out-degree at any vertex u is no more than $f(u)$. It is clear that $(E, \mathcal{I})$ is an independent system. We verify that $(E, \mathcal{I})$ is actually a matroid.

For any subset $F \subseteq E$, let $d_F^+(u)$ be the number of out-edges at u which belong to F. Then, all maximal independent sets in F have the same size,

$$\sum_{u \in V} \min\{f(u), d_F^+(u)\}.$$

Therefore, $(E, \mathcal{I})$ is a matroid. □

In a matroid, all maximal independent subsets have the same cardinality. They are called *bases*. For instance, in a graph matroid defined by a connected graph $G = (V, E)$, every base is a spanning tree of G and they all have the same size $|V| - 1$.

There is an interesting relationship between the intersection of matroids and independent systems.

Theorem 2.11 *For any independent system $(E, \mathcal{I})$, there exist a finite number of matroids $(E, \mathcal{I}_i)$, $1 \le i \le k$, such that $\mathcal{I} = \bigcap_{i=1}^{k} \mathcal{I}_i$.*

Proof. Let $C_1, \ldots, C_k$ be all minimal dependent sets of $(E, \mathcal{I})$ (i.e, they are the minimal sets among $\{F \mid F \subseteq E, F \notin \mathcal{I}\}$). For each $i \in \{1, 2, \ldots, k\}$, define

$$\mathcal{I}_i = \{F \subseteq E \mid C_i \not\subseteq F\}.$$

Then it is not hard to verify that $\mathcal{I} = \bigcap_{i=1}^{k} \mathcal{I}_i$. We next show that each $(E, \mathcal{I}_i)$ is a matroid.

It is easy to see that $(E, \mathcal{I}_i)$ is an independent system. Thus, it suffices to show that condition (I_2) holds for $(E, \mathcal{I}_i)$. Consider $F \subseteq E$. If $C_i \not\subseteq F$, then F contains a unique maximal independent set, which is itself. If $C_i \subseteq F$, then every maximal independent subset of F is equal to $F \setminus \{u\}$ for some $u \in C_i$ and hence has size $|F| - 1$. □

Theorem 2.12 *Suppose the independent system $(E, \mathcal{I})$ is the intersection of k matroids $(E, \mathcal{I}_i)$, $1 \le i \le k$; that is, $\mathcal{I} = \bigcap_{i=1}^{k} \mathcal{I}_i$. Then*

$$\max_{F \subseteq E} \frac{v(F)}{u(F)} \leq k,$$

where $u(F)$ and $v(F)$ are the two functions defined in (2.1).

Proof. Let $F \subseteq E$. Consider two maximal independent subsets I and J of F with respect to $(E, \mathcal{I})$. For each $1 \leq i \leq k$, let I_i be a maximal independent subset of $I \cup J$ with respect to $(E, \mathcal{I}_i)$ that contains I. [Note that I is an independent subset of $I \cup J$ with respect to $(E, \mathcal{I}_i)$, and so such a set I_i exists.] For any $e \in J \setminus I$, if $e \in \bigcap_{i=1}^{k} (I_i \setminus I)$, then $I \cup \{e\} \in \bigcap_{i=1}^{k} \mathcal{I}_i = \mathcal{I}$, contradicting the maximality of I. Hence, e occurs in at most $k-1$ different subsets $I_i \setminus I$. It follows that

$$\sum_{i=1}^{k} |I_i| - k|I| = \sum_{i=1}^{k} |I_i \setminus I| \leq (k-1)|J \setminus I| \leq (k-1)|J|,$$

or

$$\sum_{i=1}^{k} |I_i| \leq k|I| + (k-1)|J|.$$

Now, for each $1 \leq i \leq k$, let J_i be a maximal independent subset of $I \cup J$ with respect to $(E, \mathcal{I}_i)$ that contains J. Since, for each $1 \leq i \leq k$, $(E, \mathcal{I}_i)$ is a matroid, we must have $|I_i| = |J_i|$. In addition, for every $1 \leq i \leq k$, $|J| \leq |J_i|$. Therefore, we get

$$k|J| \leq \sum_{i=1}^{k} |J_i| = \sum_{i=1}^{k} |I_i| \leq k|I| + (k-1)|J|.$$

It follows that $|J| \leq k|I|$. □

Example 2.13 Consider the independent system $(E, \mathcal{I})$ for MAX-DHP defined in Section 2.1. Based on the analysis in the proof of Lemma 2.4 and Examples 2.9 and 2.10, we can see that $\mathcal{I}$ is actually the intersection of the following three matroids:

(1) The family $\mathcal{I}_1$ of all subgraphs with out-degree at most 1 at each vertex;

(2) The family $\mathcal{I}_2$ of all subgraphs with in-degree at most 1 at each vertex; and

(3) The family $\mathcal{I}_3$ of all subgraphs that do not contain a cycle when the edge direction is ignored.

Thus, Theorem 2.5 can also be derived from Theorem 2.12.

On the other hand, for the independent system $(E, \mathcal{I})$ for MAX-HC defined in Section 2.1, the analysis in the proof of Lemma 2.2 uses a more complicated counting argument and does not yield the simple property that $(E, \mathcal{I})$ is the intersection of two matroids. In fact, it can be proved that $(E, \mathcal{I})$ is *not* the intersection of two matroids. We remark that, in general, the problem MAX-ISS for an independent system that is the intersection of two matroids can often be solved in polynomial time. □

Example 2.14 Let X, Y, Z be three sets. We say two elements (x_1, y_1, z_1) and (x_2, y_2, z_2) in $X \times Y \times Z$ are *disjoint* if $x_1 \neq x_2$, $y_1 \neq y_2$, and $z_1 \neq z_2$. Consider the following problem:

> MAXIMUM 3-DIMENSIONAL MATCHING (MAX-3DM): Given three disjoint sets X, Y, Z and a nonnegative weight function c on all triples in $X \times Y \times Z$, find a collection $\mathcal{F}$ of disjoint triples with the maximum total weight.

For given sets X, Y, and Z, let $E = X \times Y \times Z$. Also, let $\mathcal{I}_X$ ($\mathcal{I}_Y$, $\mathcal{I}_Z$) be the family of subsets A of E such that no two triples in any subset share an element in X (Y, Z, respectively). Then $(E, \mathcal{I}_X)$, $(E, \mathcal{I}_Y)$, and $(E, \mathcal{I}_Z)$ are three matroids and MAX-3DM is just the problem of finding the maximum-weight intersection of these three matroids. By Theorem 2.12, we see that Algorithm 2.A is a polynomial-time 3-approximation for MAX-3DM. □

2.3 Quadrilateral Condition on Cost Functions

Theorem 2.7 gives us a tight relationship between matroids and the optimality of greedy algorithms. It is interesting to point out that this tight relationship holds with respect to *arbitrary* nonnegative objective functions c. That is, if $(E, \mathcal{I})$ is a matroid, then the greedy algorithm will find optimal solutions for all objective functions c. On the other hand, if $(E, \mathcal{I})$ is not a matroid, then the greedy algorithm may still produce an optimal solution, but the optimality must depend on some specific properties of the objective functions. In this section, we present such a property.

Consider a directed graph $G = (V, E)$ and a cost function $c : E \to \mathbb{R}$. We say (G, c) satisfies the *quadrilateral condition* if, for any four vertices u, v, u', v' in V,

$$\begin{aligned} c(u, v) \geq \max\{c(u, v'), c(u', v)\} \\ \Longrightarrow c(u, v) + c(u', v') \geq c(u, v') + c(u', v). \end{aligned}$$

The quadrilateral condition is quite useful in the analysis of greedy algorithms. The following are some examples.

Let $G = (V_1, V_2, E)$ be a complete bipartite graph with $|V_1| = |V_2|$. Let $\mathcal{I}$ be the family of all matchings (recall that a *matching* of a graph is a set of edges that do not share any common vertex). Clearly, $(E, \mathcal{I})$ is an independent system. It is, however, not a matroid. In fact, for some subgraphs of G, maximal matchings may have different cardinalities (although all maximal matchings for G always have the same cardinality). A maximal matching in the bipartite graph is called an *assignment*.

> MAXIMUM ASSIGNMENT (MAX-ASSIGN): Given a complete bipartite graph $G = (V_1, V_2, E)$ with $|V_1| = |V_2|$, and an edge weight function $c : E \to \mathbb{R}^+$, find a maximum-weight assignment.

Theorem 2.15 *If the weight function c satisfies the* quadrilateral condition *for all $u, u' \in V_1$ and $v, v' \in V_2$, then* Algorithm 2.A *produces an optimal solution for the instance (G, c) of* MAX-ASSIGN.

Proof. Assume that $V_1 = \{u_1, u_2, \ldots, u_n\}$ and $V_2 = \{v_1, v_2, \ldots, v_n\}$. Also, assume, without loss of generality, that $M = \{(u_i, v_i) \mid i = 1, 2, \ldots, n\}$ is the assignment found by Algorithm 2.A, in the order of (u_1, v_1), $(u_2, v_2), \ldots, (u_n, v_n)$. We claim that there must be an optimal assignment that contains the edge (u_1, v_1): Let $M^* \subseteq E$ be an arbitrary optimal solution. If the edge (u_1, v_1) is not in M^*, then M^* must have two edges (u_1, v') and (u', v_1), where $v' \neq v_1$ and $u' \neq u_1$. From the greedy strategy of Algorithm 2.A, we know that $c(u_1, v_1) \geq \max\{c(u_1, v'), c(u', v_1)\}$. Therefore, by the quadrilateral condition,

$$c(u_1, v_1) + c(u', v') \geq c(u_1, v') + c(u', v_1).$$

This means that replacing edges (u_1, v') and (u', v_1) in M^* by (u_1, v_1) and (u', v') does not decrease the total weight of the assignment. This completes the proof of the claim.

Using the same argument, we can prove that for each $i = 1, 2, \ldots, n$, there exists an optimal assignment that contains all edges $(u_1, v_1), \ldots, (u_i, v_i)$. Thus, M is actually an optimal solution. □

Next, let us come back to the problem MAX-DHP.

Theorem 2.16 *For the problem* MAX-DHP *restricted to the graphs with distance functions satisfying the quadrilateral condition, the greedy* Algorithm 2.A *is a polynomial-time* 2*-approximation.*

Proof. Assume that $G = (V, E)$ is a directed graph, and $c : E \to \mathbb{R}^+$ is the distance function. Let $n = |V|$. Let $e_1, e_2, \ldots, e_{n-1}$ be the edges selected by Algorithm 2.A into the solution set H, in the order of their selection into H. They are, hence, in nonincreasing order of their length. For each $i = 1, 2, \ldots, n-1$, let P_i be a longest simple path in G that contains edges $e_1, e_2, \ldots, e_i$, and let $Q_i = P_i - \{e_1, e_2, \ldots, e_i\}$. In particular, $Q_0 = P_0$ is an optimal solution, and $Q_{n-1} = \emptyset$. For any set T of edges in G, we write $c(T)$ to denote the total length of edges in T. We claim that for $i = 1, 2, \ldots, n-1$,

$$c(Q_{i-1}) \leq c(Q_i) + 2c(e_i).$$

To prove the claim, let us consider the relationship between P_{i-1} and P_i. If $P_{i-1} = P_i$, then $Q_{i-1} = Q_i \cup \{e_i\}$, and so

$$c(Q_{i-1}) = c(Q_i) + c(e_i) \leq c(Q_i) + 2c(e_i).$$

If $P_{i-1} \neq P_i$, then we must have $e_i \notin P_{i-1}$. Assume that $e_i = (u, v)$. To add e_i to P_{i-1} to form a simple path P_i, we must remove up to three edges from P_{i-1} (and add e_i and some new edges):

(1) The edge in P_{i-1} that begins with u;

(2) The edge in P_{i-1} that ends with v; and

(3) An edge in the path from v to u if P_{i-1} contains such a subpath.

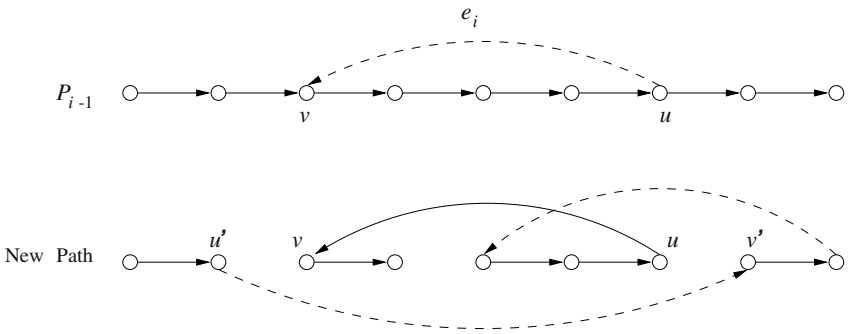


Figure 2.3: From path P_{i-1} to a new path.

In addition, these edges are all in $Q_{i-1} \setminus \{e_i\}$. Figure 2.3 shows an example of this process.

From the greedy strategy of the algorithm, we know that $c(e_i) \geq c(e)$ for any edge $e \in Q_{i-1}$. So, the total length of the edges removed is at most $3c(e_i)$. We consider two cases:

Case 1. We may form a new path passing through $e_1, \ldots, e_i$ from P_{i-1} by removing at most two edges, say, e'_j and e'_k. Then, $c((P_{i-1} \setminus \{e'_j, e'_k\}) \cup \{e_i\}) \leq c(P_i)$. Hence,

$$c(Q_{i-1}) \leq c(Q_i) + c(\{e'_j, e'_k\}) \leq c(Q_i) + 2c(e_i).$$

Case 2. We must remove three edges from P_{i-1} to form a new path passing through $e_1, e_2, \ldots, e_i$. As discussed above, these three edges must be (u, v'), (u', v), for some $u', v' \in V$, and an edge e in the subpath from v to u in P_{i-1}, and u, v, u', and v' are all distinct. This means that P_{i-1} has a subpath from u' to v', which contains these three edges. Thus, after deleting (u, v'), (u', v), and e, we can add edge (u', v') to form a new path (cf. Figure 2.3). Therefore, we have

$$\begin{aligned} c(Q_i) &\geq c(Q_{i-1}) - c(\{(u', v), e, (u, v')\}) + c(u', v') \\ &\geq c(Q_{i-1}) - c(e) - c(u, v) \\ &\geq c(Q_{i-1}) - 2c(e_i), \end{aligned}$$

where the second inequality follows from the quadrilateral condition on u, v, u', and v' and the fact that $c(u, v) \geq c(e')$ for all $e' \in Q_{i-1}$. This completes the proof of the claim.

Now, we note that $Q_{n-1} = \emptyset$, and so $c(Q_{n-1}) = 0$. Thus, we have

$$\begin{aligned} c(P_0) = c(Q_0) &\leq c(Q_1) + 2c(e_1) \\ &\leq c(Q_2) + 2c(e_1) + 2c(e_2) \\ &\leq \cdots \leq c(Q_{n-1}) + 2\sum_{i=1}^{n-1} c(e_i) = 2c(H). \end{aligned}$$

□

The quadrilateral condition sometimes holds naturally. The following is an example.

Recall that a *(character) string* is a sequence of characters from a finite alphabet Σ. We say a string s is a *superstring* of t, or t is a *substring* of s, if there exist strings u, v such that $s = utv$. If u is empty, we say t is a *prefix* of s, and if v is empty, then we say t is a *suffix* of s. The length of a string s is the number of characters in s, and is denoted by $|s|$.

> SHORTEST SUPERSTRING (SS): Given a set of strings $S = \{s_1, s_2, \ldots, s_n\}$ in which no string s_i is a substring of any other string $s_j, j \neq i$, find the shortest string s^* that contains all strings in S as substrings.

The problem SS has important applications in computational biology and data compression.

A string v is called an *overlap* of string s with respect to string t if v is both a suffix of s and a prefix of t, that is, if $s = uv$ and $t = vw$ for some strings u and w. We note that the overlap string may be an empty string. Also, the notion of overlap strings is not symmetric. That is, an overlap of s with respect to t may not be an overlap of t with respect to s. For any two strings s and t, we write $ov(s, t)$ to denote the longest overlap of s with respect to t.

To find an approximation algorithm for SS, we can transform the problem SS into the problem MAX-DHP: First, for any set $S = \{s_1, s_2, \ldots, s_n\}$ of strings, we define the *overlap graph* $G(S) = (S, E)$ to be the complete directed graph on the vertex set S, with all self-loops removed. For each edge (s_i, s_j) in E, we let its length be $c(s_i, s_j) = |ov(s_i, s_j)|$.

Suppose that s^* is a shortest superstring for S and that $s_1, s_2, \ldots, s_n$ are the strings in S in the order of occurrence from left to right in s^*. Then, for each $i = 1, \ldots, n-1$, s_i and s_{i+1} must have the maximal overlap in s^* for, otherwise, s^* could be shortened and would not be the shortest superstring. It is not hard to verify that the sequence $(s_1, s_2, \ldots, s_n)$ forms a directed Hamiltonian path H in the overlap graph $G(S)$, whose total edge length, denoted by $c(H)$, is equal to the sum of the total length of all overlap strings in s^*:

$$c(H) = \sum_{i=1}^{n-1} |ov(s_i, s_{i+1})|.$$

Next, consider an arbitrary directed Hamiltonian path $H = (s_{h(1)}, s_{h(2)}, \ldots, s_{h(n)})$ in $G(S)$. We can construct a superstring for S from H as follows: For each $i = 1, 2, \ldots, n-1$, let z_i be the prefix of $s_{h(i)}$ such that $s_{h(i)} = z_i \cdot ov(s_{h(i)}, s_{h(i+1)})$. Then, define $p(H) = z_1 z_2 \cdots z_{n-1} s_{h(n)}$. It is easy to check that $p(H)$ is a superstring of all $s_{h(i)}$, for $i = 1, 2, \ldots, n$ (cf. Figure 2.4). Clearly,

$$\begin{aligned} p(H)| &= \sum_{i=1}^{n-1} |z_i| + |s_{h(n)}| \\ &= \sum_{i=1}^{n-1} (|s_{h(i)}| - |ov(s_{h(i)}, s_{h(i+1)})|) + |s_{h(n)}| \end{aligned}$$

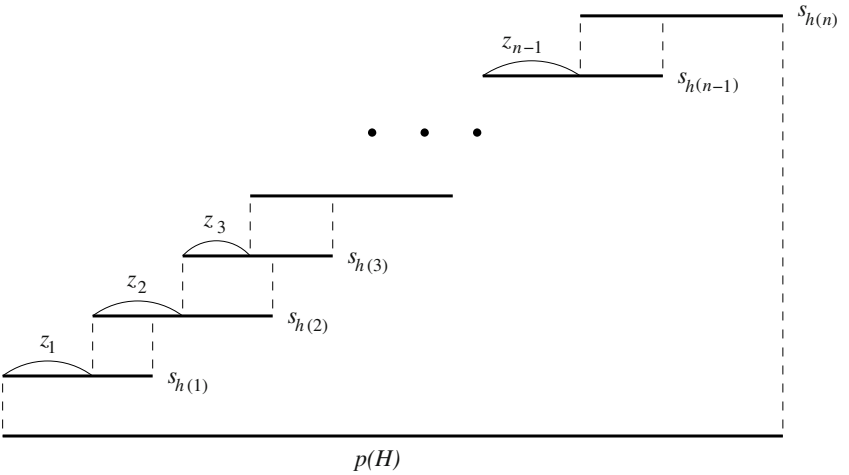


Figure 2.4: A superstring obtained from a Hamiltonian path.

$$= \sum_{i=1}^{n} |s_{h(i)}| - \sum_{i=1}^{n-1} |ov(s_{h(i)}, s_{h(i+1)})| = \sum_{i=1}^{n} |s_i| - c(H).$$

That is, the length of $p(H)$ equals the total length of the strings in S minus the total edge length of the path H. It follows that the string $p(H)$ generated from a longest directed Hamiltonian path H is a shortest superstring of S, and vice versa.

Theorem 2.17 *If H is a longest directed Hamiltonian path in the overlap graph $G(S)$, then the string $p(H)$ is a shortest superstring for S. Conversely, if s^* is a shortest superstring for S, then $s^* = p(H)$ for some longest directed Hamiltonian path H in $G(S)$.*

From this relationship, we can convert Algorithm 2.A into an approximation algorithm for the problem SS.

Algorithm 2.B (*Greedy Algorithm for* SS)
Input: A set $S = \{s_1, s_2, \ldots, s_n\}$ of strings.

(1) Set $G \leftarrow \{s_1, s_2, \ldots, s_n\}$.

(2) **While** $|G| > 1$ **do**

select s_i, s_j in G with the maximum $|ov(s_i, s_j)|$;
let $s_i \leftarrow s_i u$, where $s_j = ov(s_i, s_j)u$;
$G \leftarrow G \setminus \{s_j\}$.

(3) Output the only string s_G left in G. ■

Tarhio and Ukkonen [1988] and Turner [1989] noticed independently that the overlap graph $G(S)$ satisfies the quadrilateral condition.

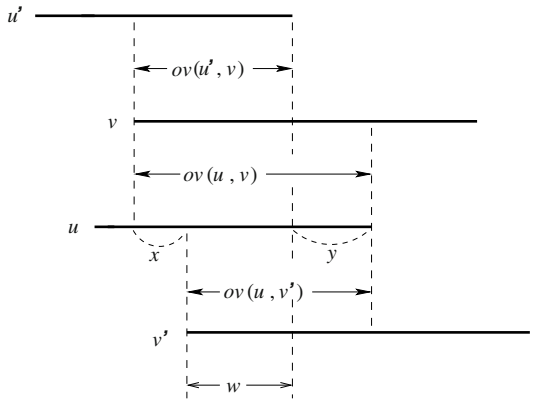


Figure 2.5: Overlaps among four strings.

Lemma 2.18 *Let $G(S)$ be the overlap graph of a set S of strings. Let u, v, u', and v' be four distinct strings in S. If $|ov(u,v)| \geq \max\{|ov(u,v')|, |ov(u',v)|\}$, then*

$$|ov(u,v)| + |ov(u',v')| \geq |ov(u,v')| + |ov(u',v)|.$$

Proof. The proof is trivial when $|ov(u,v)| \geq |ov(u,v')| + |ov(u',v)|$. Thus, we may assume that $|ov(u,v)| < |ov(u,v')| + |ov(u',v)|$.

Since both $ov(u,v)$ and $ov(u',v)$ are prefixes of v, $|ov(u',v)| \leq |ov(u,v)|$ implies that $ov(u',v)$ is a prefix of $ov(u,v)$. Similarly, we get that $ov(u,v')$ is a postfix of $ov(u,v)$ (see Figure 2.5). Because $|ov(u,v)| < |ov(u,v')| + |ov(u',v)|$, we know that the overlap of $ov(u',v)$ with respect to $ov(u,v')$ is not empty. Let $w = ov(ov(u',v), ov(u,v'))$. Then, we have $ov(u,v) = xwy$, $ov(u',v) = xw$ and $ov(u,v') = wy$ for some strings x and y (cf. Figure 2.5). That is, w is an overlap of u' with respect to v'. It follows that

$$|ov(u',v')| \geq |w| = |ov(u,v')| + |ov(u',v)| - |ov(u,v)|. \qquad \square$$

Theorem 2.19 *Let s^* be a shortest superstring for S. Let $\|S\|$ be the total length of strings in S. Then*

$$\|S\| - |s^*| \leq 2(\|S\| - s_G),$$

where s_G is the superstring generated by Algorithm 2.B.

Proof. The theorem follows immediately from Lemma 2.18 and Theorem 2.16. $\square$

The following example shows that the bound on $(\|S\| - s^*)/(\|S\| - s_G)$ given in Theorem 2.19 is the best possible.

Example 2.20 Let $S = \{ab^k, b^{k+1}, b^k a\}$, where $k \geq 1$. The shortest superstring for S is $ab^{k+1}a$. However, Algorithm 2.B may generate a superstring $ab^k ab^{k+1}$ (by first merging the string ab^k with $b^k a$). Thus, for this example, we have $\|S\| - |s_G| = k$ and $\|S\| - |s^*| = 2k$. $\square$

In the above example, we also have $|s_G|/|s^*| = (2k+3)/(k+3)$. This means that the performance ratio of Algorithm 2.B cannot be better than 2. It has been conjectured that the performance ratio of Algorithm 2.B is indeed equal to 2; that is, $|s_G| \leq 2|s^*|$, while the best known result is $|s_G| \leq 4|s^*|$ [Blum et al., 1991].

In the above, we have seen a nice relationship between the problem SS and the problem MAX-DHP. This relationship can be extended to an interesting transformation from the problem SS to the traveling salesman problem TSP on directed graphs (called DIRECTED TSP).

Let $S = \{s_1, s_2, \ldots, s_n\}$ be an instance of the problem SS. Let s_{n+1} be the empty string. Consider a complete directed graph with vertex set $V = S \cup \{s_{n+1}\}$, and the distance function

$$d(s_i, s_j) = |s_i| - |ov(s_i, s_j)|,$$

for $s_i, s_j \in V$. [Note that $ov(s_{n+1}, s_i) = ov(s_i, s_{n+1}) = s_{n+1}$ for all $1 \leq i \leq n$.] It is easy to see that the shortest superstring for set S corresponds to a minimum Hamiltonian circuit with respect to the above distance function, and vice versa. Thus, a good approximation for this special case of DIRECTED TSP would also be a good approximation for the problem SS. It has also been proved that the above distance function satisfies the triangle inequality; that is, for any s_i, s_j, and s_k, with $1 \leq i, j, k \leq n+1$, $d(s_i, s_k) \leq d(s_i, s_j) + d(s_j, s_k)$ [Turner, 1989]. Based on this relationship between the two problems DIRECTED TSP and SS, we will present, in Chapter 6, a polynomial-time 3-approximation for SS, even though no constant-ratio polynomial-time approximation for DIRECTED TSP is known.

2.4 Submodular Potential Functions

In the last three sections, we have applied the notion of independent systems to study greedy algorithms. The readers may have noticed that most applications we studied were about maximization problems. While minimization and maximization look similar, the behaviors of approximation algorithms for them are quite different. In this section, we introduce a different theory for the analysis of greedy algorithms for minimization problems.

Consider a finite set E (called the *ground* set) and a function $f : 2^E \to \mathbb{Z}$, where 2^E denotes the power set of E (i.e., the family of all subsets of E). The function f is said to be *submodular* if for any two sets A and B in 2^E,

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B). \tag{2.2}$$

Example 2.21 (a) The function $f(A) = |A|$ is submodular since

$$|A| + |B| = |A \cap B| + |A \cup B|.$$

Actually, in this case, the equality always holds, and we call f a *modular* function.

(b) Let $(E, \mathcal{I})$ be a matroid. For any $A \in 2^E$, define the *rank* of A as

$$rank(A) = \max_{I \in \mathcal{I}, I \subseteq A} |I|.$$

Then, the function *rank* is a submodular function.

To see this, consider two subsets A and B of E. Let $I_{A\cap B}$ be a maximal independent subset of $A \cap B$. Let I' be a maximal independent subset in A that contains $I_{A\cap B}$ as a subset. Since all maximal independent subsets in A have the same cardinality, we know that $|I'| = rank(A)$. Next, let I'' be a maximal independent subset in $A \cup B$ that contains I' as a subset. Similarly, we have $|I''| = rank(A \cup B)$. Let $J = I'' \setminus I'$. We note that J must be a subset of B since I' is a maximal independent subset in A. Thus, $I_{A\cap B} \cup J \subseteq I'' \cap B$ is an independent subset in B. So, $|I_{A\cap B} \cup J| = |I_{A\cap B}| + |J| \leq rank(B)$. Or,

$$\begin{aligned} rank(A \cup B) &+ rank(A \cap B) - rank(A) \\ &= |I''| + |I_{A\cap B}| - |I'| = |J| + |I_{A\cap B}| \leq rank(B). \end{aligned}$$ □

Assume that f is a submodular function on subsets of E. Define

$$\Delta_D f(C) = f(C \cup D) - f(C)$$

for any subsets C and D of E; that is, $\Delta_D f(C)$ is the extra amount of f value we gain by adding D to C. Then, the submodularity property (2.2) may be expressed as

$$\Delta_D f(A \cap B) \geq \Delta_D f(B), \tag{2.3}$$

where $D = A \setminus B$. When $D = \{x\}$ is a singleton, we simply write $\Delta_x f(C)$ instead of $\Delta_{\{x\}} f(C)$.

To see the role of submodular functions in the analysis of greedy algorithms, let us study a specific problem:

> MINIMUM SET COVER (MIN-SC): Given a set S and a collection $\mathcal{C}$ of subsets of S such that $\bigcup_{C\in\mathcal{C}} C = S$, find a subcollection $\mathcal{A} \subseteq \mathcal{C}$ with the minimum cardinality such that $\bigcup_{C\in\mathcal{A}} C = S$.

For any subcollection $\mathcal{A} \subseteq \mathcal{C}$, let $\cup\mathcal{A}$ denote the union of sets in $\mathcal{A}$; i.e., $\cup\mathcal{A} = \bigcup_{C\in\mathcal{A}} C$, and define $f(\mathcal{A}) = |\cup\mathcal{A}|$. Then f is a submodular function. To see this, we verify that, for any two subcollections $\mathcal{A}$ and $\mathcal{B}$ of $\mathcal{C}$, $f(\mathcal{A}) + f(\mathcal{B}) - f(\mathcal{A} \cup \mathcal{B})$ is equal to the number of elements in both $\cup\mathcal{A}$ and $\cup\mathcal{B}$. Moreover, every element in $\cup(\mathcal{A} \cap \mathcal{B})$ must appear in both $\cup\mathcal{A}$ and $\cup\mathcal{B}$. Therefore,

$$f(\mathcal{A}) + f(\mathcal{B}) - f(\mathcal{A} \cup \mathcal{B}) \geq f(\mathcal{A} \cap \mathcal{B}).$$

A function g on 2^E is said to be *monotone increasing* if, for all $A, B \subseteq E$,

$$A \subseteq B \Longrightarrow g(A) \leq g(B).$$

It is easy to check that the above function f is monotone increasing. We can use this function f as the potential function to design a greedy approximation for MIN-SC as follows:

Algorithm 2.C (*Greedy Algorithm for* MIN-SC)
Input: A set S and a collection $\mathcal{C}$ of subsets of S.

(1) $\mathcal{A} \leftarrow \emptyset$.

(2) **While** $f(\mathcal{A}) < |S|$ **do**

Select a set $C \in \mathcal{C}$ to maximize $f(\mathcal{A} \cup \{C\})$;

Set $\mathcal{A} \leftarrow \mathcal{A} \cup \{C\}$.

(3) Output $\mathcal{A}$. ■

This approximation algorithm can be analyzed as follows:

Theorem 2.22 *Greedy* Algorithm 2.C *is a polynomial-time* $(1 + \ln \gamma)$*-approximation for* MIN-SC*, where* γ *is the maximum cardinality of a subset in the input collection* $\mathcal{C}$*.*

Proof. Let $A_1, \ldots, A_g$ be the solution found by Algorithm 2.C, in the order of their selection into the collection $\mathcal{A}$. Denote $\mathcal{A}_i = \{A_1, \ldots, A_i\}$, for $i = 0, 1, \ldots, g$. Let $C_1, C_2, \ldots, C_m$ be a minimum set cover (i.e., $m = opt$ is the number of subsets in a minimum set cover). By the greedy strategy, we know that A_{i+1} covers the maximum number of elements that are not yet covered by $\mathcal{A}_i$. Let U_i denote the set of elements in S that are not covered by $\mathcal{A}_i$. Then the total number of elements in U_i is $|U_i| = |S| - f(\mathcal{A}_i)$. The set U_i can be covered by the m subsets in the minimum set cover $\{C_1, \ldots, C_m\}$. By the pigeonhole principle, there must be a subset C_j that covers at least $(|S| - f(\mathcal{A}_i))/m$ elements in U_i. Therefore,

$$f(\mathcal{A}_{i+1}) - f(\mathcal{A}_i) \geq \frac{|S| - f(\mathcal{A}_i)}{m}. \tag{2.4}$$

Or, equivalently,

$$|S| - f(\mathcal{A}_{i+1}) \leq (|S| - f(\mathcal{A}_i)) \cdot \Big(1 - \frac{1}{m}\Big).$$

By a simple induction, we get

$$|U_i| = |S| - f(\mathcal{A}_i) \leq |S| \cdot \Big(1 - \frac{1}{m}\Big)^i \leq |S| \cdot e^{-i/m}.$$

We note that the size of U_i decreases from $|S|$ to 0, and so there must be an integer $i \in \{1, 2, \ldots, g\}$ such that $|U_{i+1}| < m \leq |U_i|$. That is, after $i + 1$ iterations of the while-loop of step (2) of Algorithm 2.C, there are at most $m - 1$ elements left uncovered, and so the greedy Algorithm 2.C will halt after at most $m - 1$ more iterations. That is, $g \leq i + m$. In addition, we have $m \leq |U_i| \leq |S|e^{-i/m}$, and so

$$i \leq m \cdot \ln\Big(\frac{|S|}{m}\Big) \leq m \cdot \ln \gamma$$

and

$$g \leq i + m \leq m(1 + \ln \gamma).$$ □

In the above, we used the pigeonhole principle to prove inequality (2.4). It may appear that the submodularity of the potential function f is not required in the proof. It is important to point out that the above proof actually used the submodularity property of f implicitly. To clarify this point, we present, in the following, an alternative proof that uses the submodularity property of f explicitly, and avoids the use of the specific meaning of f about set coverings.

Alternative Proof for (2.4). Recall that $\{C_1, \ldots, C_m\}$ is a minimum set cover. For each $j = 1, 2, \ldots, m$, let $\mathcal{C}_j = \{C_1, \ldots, C_j\}$. By the greedy strategy, we have, for each $1 \leq j \leq m$,

$$f(\mathcal{A}_{i+1}) - f(\mathcal{A}_i) = \Delta_{A_{i+1}} f(\mathcal{A}_i) \geq \Delta_{C_j} f(\mathcal{A}_i),$$

and so

$$f(\mathcal{A}_{i+1}) - f(\mathcal{A}_i) \geq \frac{1}{m} \cdot \sum_{j=1}^{m} \Delta_{C_j} f(\mathcal{A}_i).$$

On the other hand, we note that

$$|S| - f(\mathcal{A}_i) = f(\mathcal{A}_i \cup \mathcal{C}_m) - f(\mathcal{A}_i) = \sum_{j=1}^{m} \Delta_{C_j} f(\mathcal{A}_i \cup \mathcal{C}_{j-1}).$$

Therefore, to get (2.4), it suffices to have

$$\Delta_{C_j} f(\mathcal{A}_i) \geq \Delta_{C_j} f(\mathcal{A}_i \cup \mathcal{C}_{j-1}),$$

which follows from the submodularity and monotone increasing properties of the function f. □

The second proof above illustrates that the submodularity and monotone increasing properties of the potential function are sufficient conditions for inequality (2.4). In particular, for $m = 2$, inequality (2.4) is equivalent to

$$\Delta_{C_2} f(\mathcal{A}_i) \geq \Delta_{C_2} f(\mathcal{A}_i \cup C_1).$$

We will show, in the following, that this is equivalent to the condition that f is submodular and monotone increasing.

Lemma 2.23 *Let f be a submodular function on 2^E. Then, for all sets $A, C \subseteq E$,*

$$\Delta_C f(A) \leq \sum_{x \in C} \Delta_x f(A).$$

Proof. Note that if $x \in A$, then $\Delta_x f(A) = 0$. Thus, without loss of generality, we may assume that $A \cap C = \emptyset$. For any $x \in C$, set $X = A \cup \{x\}$ and $Y = A \cup (C - \{x\})$. Then, by the definition of submodular functions, we have

$$f(C \cup A) + f(A) = f(X \cup Y) + f(X \cap Y)$$
$$\leq f(X) + f(Y) = f(A \cup \{x\}) + f(A \cup (C - \{x\})).$$

It follows that

$$\Delta_C f(A) \leq \Delta_x f(A) + \Delta_{C-\{x\}} f(A).$$

The lemma can now be derived easily from this inequality. □

Lemma 2.24 *Let f be a function on all subsets of a set E. Then f is submodular if and only if, for any two subsets $A \subseteq B$ of E and any element $x \notin B$,*

$$\Delta_x f(A) \geq \Delta_x f(B). \tag{2.5}$$

Proof. From $A \subseteq B$ and $x \notin B$, we know that $(A \cup \{x\}) \cup B = B \cup \{x\}$ and $(A \cup \{x\}) \cap B = A$. Therefore, if f is submodular, then

$$f(A \cup \{x\}) + f(B) \geq f(A) + f(B \cup \{x\}).$$

That is,

$$\Delta_x f(A) \geq \Delta_x f(B).$$

Conversely, suppose (2.5) holds for all subsets $A \subseteq B$ and all $x \notin B$. Consider two arbitrary subsets A, B of E. Let $D = A \backslash B$, and assume that $D = \{x_1, \ldots, x_k\}$. Then

$$\Delta_D f(A \cap B) = \sum_{i=1}^{k} \Delta_{x_i} f((A \cap B) \cup \{x_1, \ldots, x_{i-1}\})$$
$$\geq \sum_{i=1}^{k} \Delta_{x_i} f(B \cup \{x_1, \ldots, x_{i-1}\}) = \Delta_D f(B).$$

(Note that $D = A \setminus B$, and so $x_i \notin B$ for all $i = 1, 2, \ldots, n$.) That is, inequality (2.3) holds and hence f is submodular. □

Lemma 2.25 *Let f be a function on all subsets of a set E. Then f is submodular and monotone increasing if and only if, for any two subsets $A \subseteq B$ and any element $x \in E$,*

$$\Delta_x f(A) \geq \Delta_x f(B).$$

Proof. We note that f is monotone increasing if and only if, for any subset $A \subseteq E$ and any $x \in E$, $\Delta_x f(A) \geq 0$. Now, assume that f is also submodular. Then, for any subsets $A \subseteq B \subseteq E$ and any $x \in E \setminus B$, we have, by Lemma 2.24, $\Delta_x f(A) \geq \Delta_x f(B)$; and for $x \in B$, we also have, by monotonicity of f, $\Delta_x f(A) \geq 0 = \Delta_x f(B)$.

Conversely, assume that $\Delta_x f(A) \geq \Delta_x f(B)$ for any subsets $A \subseteq B \subseteq E$ and any $x \in E$. Then, by Lemma 2.24, we know that f is submodular. In addition, set

$B = E$; we get $\Delta_x f(A) \geq \Delta_x f(E) = 0$ for all $x \in E$, which implies that f is monotone increasing. □

A submodular function is *normalized* if $f(\emptyset) = 0$. Every submodular function f can be normalized by setting $g(A) = f(A) - f(\emptyset)$. We note that if f is a normalized, monotone increasing submodular function, then $f(A) \geq 0$ for every set $A \subseteq E$. A normalized, monotone increasing, submodular function f is also called a *polymatroid function*. If f is defined on 2^E, then (E, f) is called a *polymatroid*. There are close relationships among polymatroids, matroids, and independent systems; see Exercises 2.18–2.24.

Consider a submodular function f on 2^E. Let $\Omega_f = \{C \subseteq E \mid (\forall x \in E)\ \Delta_x f(C) = 0\}$. Intuitively, Ω_f contains the *maximal sets* C under function f; that is, $f(C \cup B) = f(C)$ for all sets B.

Lemma 2.26 *Let f be a monotone increasing, submodular potential function on 2^E. Then, $\Omega_f = \{C \mid f(C) = f(E)\}$.*

Proof. If $C \in \Omega_f$, then

$$0 \leq f(E) - f(C) = \Delta_{E-C} f(C) \leq \sum_{x \in E-C} \Delta_x f(C) = 0.$$

Therefore, $f(C) = f(E)$.

Conversely, if $f(C) = f(E)$, then, for any $x \in E$, $f(C) \leq f(C \cup \{x\}) \leq f(E)$, and so $f(C) = f(C \cup \{x\})$. That is, for any $x \in E$, $\Delta_x f(C) = 0$. □

We are now ready to present a general result about greedy approximations which use a monotone increasing, submodular function as the potential function. Consider the following minimization problem.

> MINIMUM SUBMODULAR COVER (MIN-SMC): Given a finite set E, a normalized, monotone increasing, submodular function f on 2^E, and a nonnegative cost function c on E,
>
> $$\begin{aligned} &\text{minimize} && c(A) = \sum_{x \in A} c(x), \\ &\text{subject to} && A \in \Omega_f. \end{aligned}$$

This minimization problem is a general form for many problems. In most applications, the submodular function f is not given explicitly in the form of the input/output pairs, but its value at any set $A \subseteq E$ is computable in polynomial time.

Example 2.27 Consider the weighted version of the problem MIN-SC.

> MINIMUM-WEIGHT SET COVER (MIN-WSC): Given a set S, a collection $\mathcal{C}$ of subsets of S with $\cup \mathcal{C} = S$, and a weight function w on all sets $C \in \mathcal{C}$, find a set cover with the minimum total weight.

Following the discussion on MIN-SC, let the input collection $\mathcal{C}$ be the ground set, and define, for any subcollection $\mathcal{A}$ of $\mathcal{C}$, $f(\mathcal{A}) = |\cup\mathcal{A}|$. Then, f is a submodular function. Moreover, f is apparently monotone increasing. With this function f, $\Delta_C f(\mathcal{A}) = 0$ if and only if $C \subseteq \cup\mathcal{A}$. This means that a subcollection $\mathcal{A}$ belongs to Ω_f if and only if $\mathcal{A}$ is a set cover of $S = \cup\mathcal{C}$. Thus, the problem MIN-WSC is just the problem MIN-SMC with respect to this potential function f. □

Example 2.28 A *hypergraph* $H = (V, \mathcal{C})$ is a pair of sets V and $\mathcal{C}$, where $\mathcal{C}$ is a family of subsets of V. Each element in V is called a *vertex* and each subset in $\mathcal{C}$ is called an *edge* (and sometimes, to emphasize that it is an edge of a hypergraph, called a *hyperedge*). The *degree* of a vertex is the number of edges that contain the vertex.

A subset A of vertices is called a *hitting set* of the hypergraph $H = (V, \mathcal{C})$ if every edge in $\mathcal{C}$ contains at least one vertex from A. The following problem is the weighted version of MIN-HS defined in Exercise 1.15:

> MINIMUM-WEIGHT HITTING SET (MIN-WHS): Given a hypergraph $H = (V, \mathcal{C})$ and a nonnegative weight function c on vertices in V, find a hitting set $A \subseteq V$ of the minimum total weight.

Let V be the ground set, and define, for each $A \subseteq V$, $E(A)$ to be the collection of sets $C \in \mathcal{C}$ such that $C \cap A \neq \emptyset$, and let $f(A) = |E(A)|$. Then it is easy to see that $E(A \cup B) = E(A) \cup E(B)$ and $E(A \cap B) \subseteq E(A) \cap E(B)$. Thus, we have

$$\begin{aligned} |E(A)| + |E(B)| &= |E(A) \cup E(B)| + |E(A) \cap E(B)| \\ &\geq |E(A \cup B)| + |E(A \cap B)|. \end{aligned}$$

That is, function f is a submodular function. Furthermore, it is easy to check that $E(\emptyset) = \emptyset$, and if $A \subseteq B$, then $E(A) \subseteq E(B)$. Thus, f is a normalized, monotone increasing, submodular function.

Now, what is Ω_f? It is not hard to verify that $A \in \Omega_f$ if and only if A is a hitting set. Thus, the problem MIN-WHS is just the problem MIN-SMC with respect to this submodular potential function f. □

The problem MIN-SMC has a natural greedy algorithm: In each iteration, we add an element x to the solution set A to maximize the value $\Delta_x f(A)$, relative to the cost $c(x)$.

Algorithm 2.D (*Greedy Algorithm* for MIN-SMC)
Input: A finite set E, a submodular function f on 2^E, and a function $c : E \to \mathbb{R}^+$.

(1) Set $A \leftarrow \emptyset$.

(2) **While** there exists an $x \in E$ such that $\Delta_x f(A) > 0$ **do**

> select a vertex x that maximizes $\Delta_x f(A)/c(x)$;
> $A \leftarrow A \cup \{x\}$.

(3) **Return** $A_G \leftarrow A$. ■

The following theorem gives an estimation of the performance of this algorithm. We write $H(n)$ to denote the *harmonic function* $H(n) = \sum_{i=1}^{n} 1/i$. Note that $H(n) \leq 1 + \ln n$ (see Exercise 2.6).

Theorem 2.29 *Let f be a normalized, monotone increasing, submodular function. Then* Algorithm 2.D *produces an approximate solution within a factor of $H(\gamma)$ from the optimal solution to the input (E, f, c), where $\gamma = \max_{x \in E} f(\{x\})$.*

Proof. Let A be the approximate solution obtained by Algorithm 2.D. Assume that $x_1, x_2, \ldots, x_k$ are the elements of A, in the order of their selection into the set. Denote $A_i = \{x_1, x_2, \ldots, x_i\}$; in particular, $A_0 = \emptyset$. Let A^* be an optimal solution to the same instance.

For any set $B \subseteq E$, we write $c(B)$ to denote the total cost of B: $c(B) = \sum_{x \in B} c(x)$. We are going to prove that

$$c(A) \leq c(A^*) \cdot H(\gamma)$$

by a weight-decomposition counting argument. That is, we decompose the total cost $c(A)$ of the approximate solution and distribute it to the elements of the optimal solution A^* through a weight function $w(y)$ on $y \in A^*$. Then we calculate the weight decomposition according to the optimal solution A^* and show that each element $y \in A^*$ can pick up at most weight $c(y) \cdot H(\gamma)$. It follows, therefore, that $c(A^*)$ is at least $c(A)/H(\gamma)$.

In other words, we need to assign weight $w(y)$ to each element y of A^* so that it satisfies the following properties:

(a) $c(A) \leq \sum_{y \in A^*} w(y)$; and

(b) $w(y) \leq c(y) \cdot H(\gamma)$.

Property (b) implies that $\sum_{y \in A^*} w(y) \leq c(A^*)H(\gamma)$. Thus, properties (a) and (b) together establish the desired result.

First, to simplify the notation, we let $r_i = \Delta_{x_i} f(A_{i-1})$ and $z_{y,i} = \Delta_y f(A_{i-1})$. Now, we define, for each $y \in A^*$,

$$w(y) = \sum_{i=1}^{k} (z_{y,i} - z_{y,i+1}) \frac{c(x_i)}{r_i}.$$

Before we prove properties (a) and (b), we observe that

$$\sum_{i=1}^{k} (z_{y,i} - z_{y,i+1}) = z_{y,1} - z_{y,k+1} = \Delta_y f(A_0) - \Delta_y f(A_k) = f(\{y\}).$$

[In the above, $\Delta_y f(A_0) = f(\{y\})$ because f is normalized, and $\Delta_y f(A_k) = 0$ because $A_k = A \in \Omega_f$.] Therefore,

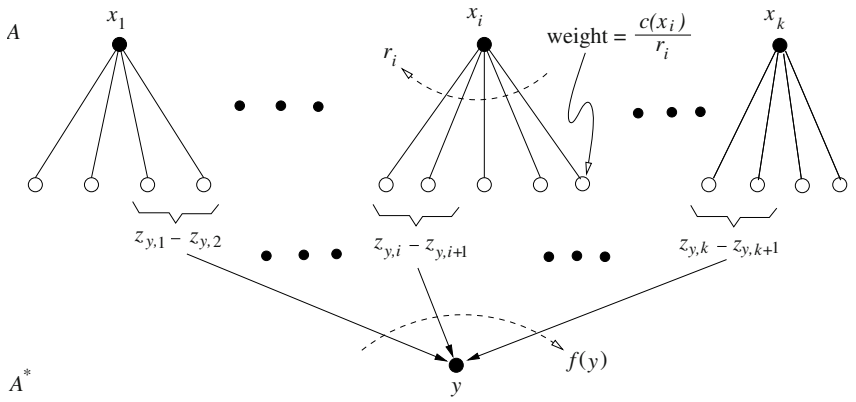


Figure 2.6: The weight decomposition.

$$\sum_{y\in A^*}\sum_{i=1}^{k}(z_{y,i}-z_{y,i+1}) = \sum_{y\in A^*} f(\{y\})$$
$$\geq f(A^*) = f(A) = \sum_{i=1}^{k}\Delta_{x_i} f(A_{i-1}) = \sum_{i=1}^{k} r_i,$$

since both A^* and A are in Ω_f. This relationship provides some intuition about how the weight-decomposition function is defined: As illustrated in Figure 2.6, we divide each element x_i into r_i parts, each of weight $c(x_i)/r_i$, so that the total weight of all parts, over all $x_i \in A$, is $c(A)$. Then each $y \in A^*$ picks up $z_{y,i} - z_{y,i+1}$ parts from the element x_i. The total number of parts picked up by y, disregarding the different weight, is $f(\{y\})$. Our goal here is to distribute part of each $x_i \in A$ to some $y \in A^*$, while each $y \in A^*$ does not take too much weight.

We now proceed to prove properties (a) and (b). For property (a), we can write weight $w(y)$ in the following form:

$$w(y) = \sum_{i=1}^{k}(z_{y,i}-z_{y,i+1})\frac{c(x_i)}{r_i}$$
$$= \frac{c(x_1)}{r_1}z_{y,1} + \sum_{i=2}^{k}\left(\frac{c(x_i)}{r_i} - \frac{c(x_{i-1})}{r_{i-1}}\right)z_{y,i}.$$

[Note that $z_{y,k+1} = \Delta_y f(A_k) = 0$.] In addition, $c(A)$ can also be expressed in a similar form:

$$c(A) = \sum_{i=1}^{k}\frac{r_i}{r_i}c(x_i) = \sum_{i=1}^{k}\left(\sum_{j=i}^{k} r_j - \sum_{j=i+1}^{k} r_j\right)\frac{c(x_i)}{r_i}$$
$$= \frac{c(x_1)}{r_1}\sum_{j=1}^{k} r_j + \sum_{i=2}^{k}\left(\frac{c(x_i)}{r_i} - \frac{c(x_{i-1})}{r_{i-1}}\right)\sum_{j=i}^{k} r_j.$$

Moreover, from the greedy strategy of Algorithm 2.D, we know that

$$\frac{r_1}{c(x_1)} \geq \frac{r_2}{c(x_2)} \geq \cdots \geq \frac{r_k}{c(x_k)};$$

or, equivalently,

$$\frac{c(x_i)}{r_i} - \frac{c(x_{i-1})}{r_{i-1}} \geq 0,$$

for all $i = 1, \ldots, k$. Thus, to prove (a), it suffices to prove that for any $i = 1, 2, \ldots, k$,

$$\sum_{j=i}^{k} r_j \leq \sum_{y \in A^*} z_{y,i}.$$

This inequality holds since, by Lemmas 2.23 and 2.26,

$$\begin{aligned}
\sum_{j=i}^{k} r_j &= \sum_{j=i}^{k} \Delta_{x_j} f(A_{j-1}) = \sum_{j=i}^{k} (f(A_j) - f(A_{j-1})) \\
&= f(A) - f(A_{i-1}) = f(A^*) - f(A_{i-1}) \\
&= f(A^* \cup A_{i-1}) - f(A_{i-1}) = \Delta_{A^*} f(A_{i-1}) \\
&\leq \sum_{y \in A^*} \Delta_y f(A_{i-1}) = \sum_{y \in A^*} z_{y,i}.
\end{aligned}$$

Next, we prove property (b). Let y be a fixed element in A^*. From the greedy strategy of Algorithm 2.D, we know that if $z_{y,i} > 0$, then

$$\frac{c(x_i)}{r_i} \leq \frac{c(y)}{z_{y,i}},$$

for all $i = 1, 2, \ldots, k$. In addition, we know from Lemma 2.25 that $z_{y,i} \geq z_{y,i+1}$. Let $\ell = \max\{i \mid 1 \leq i \leq k, z_{y,i} > 0\}$. We have

$$\begin{aligned}
w(y) &= \sum_{i=1}^{\ell} (z_{y,i} - z_{y,i+1}) \frac{c(x_i)}{r_i} \\
&\leq \sum_{i=1}^{\ell} (z_{y,i} - z_{y,i+1}) \frac{c(y)}{z_{y,i}} = c(y) \sum_{i=1}^{\ell} \frac{z_{y,i} - z_{y,i+1}}{z_{y,i}}.
\end{aligned}$$

Note that for any integers $p > q > 0$, we have

$$\frac{p-q}{p} = \sum_{j=q+1}^{p} \frac{1}{p} \leq \sum_{j=q+1}^{p} \frac{1}{j} = H(p) - H(q).$$

So, we have

$$w(y) \le c(y) \sum_{i=1}^{\ell-1} \big(H(z_{y,i}) - H(z_{y,i+1})\big) + c(y)\, H(z_{y,\ell}) = c(y)\, H(z_{y,1}).$$

Note that $z_{y,1} = f(\{y\}) \le \gamma$ for all $y \in A^*$. Therefore, we have proved property (b) and, hence, the theorem. □

2.5 Applications

Now we present some applications of the greedy Algorithm 2.D.

First, from Example 2.27, we get the upper bound for the performance ratio of the greedy algorithm for MIN-WSC immediately. More specifically, the submodular potential function f for the problem MIN-WSC is defined to be $f(\mathcal{A}) = |\cup \mathcal{A}|$. Therefore, when applied to MIN-WSC, the greedy strategy for Algorithm 2.D is to select, at each stage, the set $C \in \mathcal{C}$ with the highest value of

$$\frac{|\cup (\mathcal{A} \cup \{C\})| - |\cup \mathcal{A}|}{c(C)},$$

where $c(C)$ is the weight of set C, and add C to the solution collection $\mathcal{A}$. Also, the parameter γ in the performance ratio $H(\gamma)$ of Theorem 2.29 is equal to the maximum value of $f(\{C\}) = |C|$ over all $C \in \mathcal{C}$. Therefore, we have the following result:

Corollary 2.30 *When it is applied to the problem* MIN-WSC*,* Algorithm 2.D *is a polynomial-time $H(m)$-approximation, where m is the maximum cardinality of subsets in the input collection $\mathcal{C}$.*

From Example 2.28, we know that the function $f(A) = |E(A)|$ is monotone increasing and submodular for the problem MIN-WHS. With respect to this potential function f, Algorithm 2.D selects, at each stage, the element $x \in S$ with the highest value of

$$\frac{|E(A \cup \{x\})| - |E(A)|}{c(x)},$$

and adds x to the solution set A. We note that in the setting of the problem MIN-WHS, the parameter γ in the performance ratio $H(\gamma)$ of Theorem 2.29 is just the maximum degree over all vertices. So, we get the following result:

Corollary 2.31 *When it is applied to the problem* MIN-WHS*,* Algorithm 2.D *is a polynomial-time $H(\delta)$-approximation, where δ is the maximum degree of a vertex in the input hypergraph.*

Note that if all edges in the input hypergraph $H = (V, \mathcal{C})$ have exactly two elements, then this subproblem of MIN-WHS is actually the weighted version of the vertex cover problem MIN-VC (see Exercise 1.10).

> MINIMUM-WEIGHT VERTEX COVER (MIN-WVC): Given a graph $G = (V, E)$, with a nonnegative weight function $c : V \to \mathbb{R}^+$, find a vertex cover of the minimum total weight.

We prove that the bound $H(\delta)$ of Corollary 2.31 is actually tight, even for the nonweighted version of MIN-VC on bipartite graphs.

Theorem 2.32 *For any $n \geq 1$, there exists a bipartite graph G with degree at most n and a minimum vertex cover of size $n!$ such that* Algorithm 2.D *produces a vertex cover of size $H(n) \cdot (n!)$ on graph G.*

Proof. Let $V_1, V_{2,1}, V_{2,2}, \ldots, V_{2,n}$ be $n+1$ pairwisely disjoint sets of size $|V_1| = n!$ and $|V_{2,i}| = n!/i$, for each $i = 1, 2, \ldots, n$. The bipartite graph G has the vertex sets V_1 and $V_2 = \bigcup_{i=1}^{n} V_{2,i}$. To define the edges in G, we perform the following process for each $1 \leq i \leq n$: We partition V_1 into $n!/i$ disjoint subsets, each of size i, and build a one-to-one correspondence between these $n!/i$ subsets and $n!/i$ vertices in $V_{2,i}$. Then, for each subset A of V_1, we connect every vertex in A to the vertex in $V_{2,i}$ that corresponds to subset A.

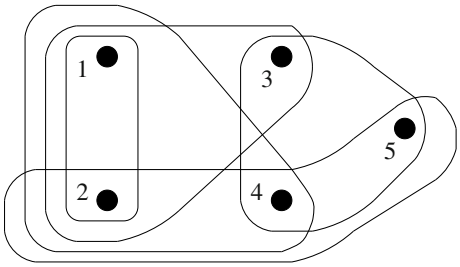
Thus, in the bipartite graph G, each vertex in V_1 has degree n and each vertex in $V_{2,i}$ has degree $i \leq n$. Clearly, V_1 is a minimum hitting set, which has size $n!$. However, the greedy Algorithm 2.D on graph G may produce V_2 as the hitting set, which has size $\sum_{i=1}^{n} (n!)/i = H(n) \cdot (n!)$. □

The above result indicates that Algorithm 2.D is not a good approximation for the nonweighted MIN-VC, as MIN-VC actually has a polynomial-time 2-approximation, and MIN-VC in bipartite graphs can be solved in polynomial time (see Exercise 1.10). On the other hand, Algorithm 2.D is probably the best approximation for the nonweighted hitting set problem, unless certain complexity hierarchies collapse (see Historical Notes).

Our next example is the problem of subset interconnection design. Recall that for any graph $G = (V, E)$ and any set $S \subseteq V$, $G|_S$ denotes the subgraph of G induced by set S; i.e., $G|_S$ is the graph with vertex set S and edge set $E|_S = \{\{x, y\} \in E \mid x, y \in S\}$. For any subsets $S_1, S_2, \ldots, S_m$ of V, we say a subgraph $H = (V, F)$ of G is a *feasible graph* for $S_1, S_2, \ldots, S_m$ if, for each $i = 1, 2, \ldots, m$, the subgraph $H|_{S_i}$ induced by S_i is connected.

> WEIGHTED SUBSET INTERCONNECTION DESIGN (WSID): Given a complete graph $G = (V, E)$ with a nonnegative edge weight function $c : E \to \mathbb{R}^+$, and m vertex subsets $S_1, S_2, \ldots, S_m \subseteq V$, find a feasible subgraph $H = (V, F)$ for $S_1, S_2, \ldots, S_m$, with the minimum total edge weight.

Example 2.33 Let $V = \{v_1, v_2, \ldots, v_5\}$, and consider the five subsets $S_1 = \{v_1, v_2\}$, $S_2 = \{v_1, v_2, v_3\}$, $S_3 = \{v_3, v_4, v_5\}$, $S_4 = \{v_1, v_2, v_4\}$, and $S_5 = \{v_2, v_4, v_5\}$. These subsets form a hypergraph on V, as shown in Figure 2.7, together with a cost function c. Figure 2.8 shows two feasible graphs for these subsets. With respect to the cost function c given in Figure 2.7, the graph in Figure 2.8(b) is a minimum-cost feasible graph. □



$c(i,j)$	1	2	3	4	5
1		5	6	7	8
2			5	6	7
3				5	6
4					5
5					

Figure 2.7: A hypergraph and its cost function.

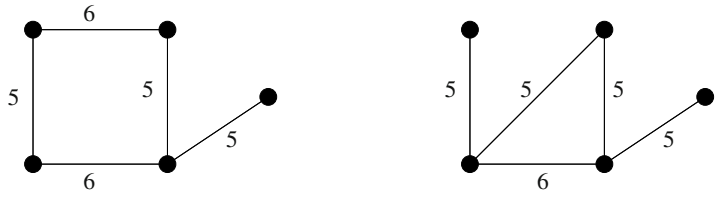


Figure 2.8: Feasible graphs for the input of Figure 2.7.

In the following, we define a submodular function r on subsets of the edge set E. Consider the graph matroid of the induced subgraph $G|_{S_i} = (V, E_i)$ (see Example 2.9), where $E_i = E|_{S_i}$. In this graph matroid, a set $I \subseteq E_i$ is an independent subset if (S_i, I) is an acyclic subgraph of $G|_{S_i}$. Let r_i be the rank function of the graph matroid of graph $G|_{S_i}$ (see Example 2.21(b)). That is, for any $A \subseteq E$, $r_i(A) =$ the size of the largest edge set $I \subseteq A \cap E_i$ such that (S_i, I) is an acyclic subgraph of $G|_{S_i}$. Equivalently,

$$r_i(A) = |S_i| - \text{the number of connected components of the graph } (S_i, A \cap E_i).$$

By Example 2.21(b), r_i is a submodular function.

Now, define $r(A) = \sum_{i=1}^{m} r_i(A)$. Note that the sum of submodular functions is submodular. Therefore, r is a submodular function. Furthermore, it is not hard to check that r is monotone increasing and normalized.

For this submodular function r, the set Ω_r is the collection of sets $A \subseteq E$ such that $r(A \cup \{e\}) = r(A)$ for all edges e in E. It is not hard to see that Ω_r is just the set of all feasible graphs. Thus, the problem WSID is actually the minimization problem MIN-SMC with respect to the submodular potential function r. So, Algorithm 2.D and Theorem 2.29 can be applied to it.

To be more precise, the greedy criterion of Algorithm 2.D for the problem WSID is to select, at each stage, an edge $\{e\}$ with the maximum ratio

$$\frac{r(F \cup \{e\}) - r(F)}{c(e)}$$

and add it to the solution edge set F. What is the value $r(F \cup \{e\}) - r(F)$? It is the number of indices $i \in \{1, 2, \ldots, m\}$ such that edge e connects two distinct connected components of the graph $G|_{F \cap S_i}$.

Also, the parameter γ of Theorem 2.29 is equal to the maximum value of $r(\{e\})$, which is the maximum number of indices $i \in \{1, 2, \ldots, m\}$ such that S_i contains the two endpoints of e.

Corollary 2.34 *When it is applied to the problem* WSID*,* Algorithm 2.D *is a polynomial-time $H(K)$-approximation, where K is the maximum number of induced subgraphs $G|_{S_i}$ that share a common edge.*

It is known that for $0 < \rho < 1$, the problem WSID has no polynomial-time approximation within a factor of $\rho \ln n$ from the optimal solution unless every **NP**-complete problem is solvable in deterministic time $O(n^{polylog n})$[1] (this condition is weaker than **NP** = **P** but is still considered not likely to be true).

For a connected graph $G = (V, E)$, we say a subset $C \subseteq V$ is a *connected vertex cover* if C is a vertex cover for G and the induced subgraph $G|_C$ is connected. Consider the following problem:

> MINIMUM-WEIGHT CONNECTED VERTEX COVER (MIN-WCVC):
> Given a connected graph $G = (V, E)$ and a nonnegative vertex weight function $c : V \to \mathbb{R}^+$, find a connected vertex cover with the minimum total weight.

For a graph $G = (V, E)$ and a subset $C \subseteq V$, let $g(C)$ be the number of edges in E that are not covered by C, and $h(C)$ the number of connected components of $G|_C$. Define $p(C) = |E| - g(C) - h(C)$. Clearly, $p(\emptyset) = |E| - g(\emptyset) - h(\emptyset) = 0$.

We are going to prove that p is a monotone increasing, submodular function, using a new characterization of submodular functions. In the following, we write $\Delta_x \Delta_y f(A)$ to denote $\Delta_y f(A \cup \{x\}) - \Delta_y f(A)$. For the proofs of the following two lemmas, see Exercise 2.14.

Lemma 2.35 *Let f be a function on 2^E. Then f is submodular if and only if for any $A \subseteq E$ and any two distinct elements $x, y \notin A$,*

$$\Delta_x \Delta_y f(A) \leq 0.$$

Lemma 2.36 *Let f be a function on 2^E. Then f is monotone increasing and submodular if and only if for any $A \subseteq E$ and $x, y \in E$,*

$$\Delta_x \Delta_y f(A) \leq 0.$$

[1] The notation *polylog* n denotes the class of functions $(\log n)^k$, for all $k \geq 1$.

Now, we apply this characterization to show that p is a monotone increasing, submodular function.

Lemma 2.37 *p is monotone increasing and submodular.*

Proof. Consider a vertex subset C and a vertex $u \notin C$. Then $\Delta_u p(C) = -\Delta_u g(C) - \Delta_u h(C)$. We observe that $-\Delta_u g(C)$ is just the number of edges incident on u in graph G that are not covered by C. It follows that $-\Delta_u g(C) = |N(u) \setminus C|$, where $N(u)$ is the set of vertices in G that are adjacent to u. Moreover, $-\Delta_u h(C)$ is equal to the number of connected components in $G|_C$ that are adjacent to u *minus* 1. Therefore, we always have $-\Delta_u g(C) \geq 0$ and $-\Delta_u h(C) \geq -1$.

By Lemma 2.36, it is sufficient to prove that for any vertex subset C and two vertices u and v,

$$\Delta_v \Delta_u p(C) \leq 0.$$

Note that if $u \in C$, then both $\Delta_u p(C \cup \{v\})$ and $\Delta_u p(C)$ are equal to 0, and hence $\Delta_v \Delta_u p(C) = 0$. Also, if $v \in C$, then we have $\Delta_u p(C \cup \{v\}) = \Delta_u p(C)$, and hence $\Delta_v \Delta_u p(C) = 0$. Thus, we may assume that neither u nor v belongs to C.

We consider three cases.

Case 1: $u = v$. Since $\Delta_u p(C \cup \{v\}) = 0$, it suffices to show $\Delta_u p(C) \geq 0$. If $C \cap N(u) = \emptyset$, then $-\Delta_u g(C) = \deg(u)$ and $\Delta_u h(C) = -1$, which implies that $\Delta_u p(C) = \deg(u) - 1 \geq 0$, because G is connected and so $\deg(u)$ is at least 1. If $C \cap N(u) \neq \emptyset$, then u is adjacent to at least one connected component of $G|_C$ and hence $-\Delta_u h(C) \geq 0$, which also implies that $\Delta_u p(C) \geq 0$.

Case 2: $u \neq v$ and u is not adjacent to v. Then $N(u) \setminus (C \cup \{v\}) = N(u) \setminus C$, and hence $-\Delta_u g(C \cup \{v\}) = -\Delta_u g(C)$. Consider an arbitrary connected component of $G|_{C \cup \{v\}}$ that is adjacent to u. If it does not contain v, then it is also a connected component of $G|_C$ adjacent to u. If it contains v, then it must contain at least one connected component of $G|_C$ adjacent to u. Thus, the number of connected components of $G|_{C \cup \{v\}}$ adjacent to u is no more than the number of connected components of $G|_C$ adjacent to u; that is, $-\Delta_u h(C \cup \{v\}) \leq -\Delta_u h(C)$. So $\Delta_u p(C \cup \{v\}) \leq \Delta_u p(C)$.

Case 3: $u \neq v$ but u is adjacent to v. Then $N(u) \setminus (C \cup \{v\}) = (N(u) \setminus C) \setminus \{v\}$, and hence $-\Delta_u g(C \cup \{v\}) = -\Delta_u g(C) - 1$. Also, among all connected components of $G|_{C \cup \{v\}}$ that are adjacent to u, exactly one contains v and all others are connected components of $G|_C$ adjacent to u. Hence, $-\Delta_u h(C \cup \{v\}) \leq -\Delta_u h(C) + 1$. Therefore, $\Delta_u p(C \cup \{v\}) \leq \Delta_u p(C)$. □

It can be verified that with respect to this submodular function p, the set Ω_p is exactly the collection of connected vertex covers of G.

Lemma 2.38 *Let $G = (V, E)$ be a connected graph with at least three vertices. For any subset $C \subseteq V$, C is a connected vertex cover if and only if, for any vertex $x \in V$, $\Delta_x p(C) = 0$.*

Proof. If C is a connected vertex cover, then it is clear that $p(C) = |E| - g(C) - h(C) = |E| - 0 - 1 = |E| - 1$, reaching the maximum value of p.

Conversely, suppose that for any vertex $x \in V$, $\Delta_x p(C) = 0$. It is clear that $C \neq \emptyset$, for otherwise we can find a vertex $x \in V$ of degree ≥ 2 and get $\Delta_x p(C) = -\Delta_x g(\emptyset) - \Delta_x h(\emptyset) \geq 2 - 1 = 1$. Now, assume, for the sake of contradiction, that C is not a connected vertex cover. Let $B = \{x \in V \mid x \text{ is adjacent to some } v \in C\}$, and $A = V \setminus (B \cup C)$. Consider two cases.

Case 1: There exists an edge in E that is not covered by C. Then there must be an edge e in E not covered by C such that one of its endpoints x is in B (otherwise, A forms a nonempty connected component of G, contradicting the assumption that G is connected). Now, we note that $C \cup \{x\}$ covers at least one extra edge e than C, and so $-g(C \cup \{x\}) > -g(C)$. In addition, since x is in B and is adjacent to at least one vertex in C, adding x to C does not increase the number of connected components. Therefore, $-h(C \cup \{x\}) \geq -h(C)$. Together, we get $\Delta_x p(C) > 0$, which is a contradiction.

Case 2: C covers every edge, but $G|_C$ is not connected. Since G is connected, there must be a path in G connecting two connected components of $G|_C$. Furthermore, such a shortest path must contain exactly two edges $\{u, x\}$ and $\{x, v\}$ with $u, v \in C$ and $x \in B$, for otherwise it would contain an edge whose two endpoints are not in C. But then we have $-h(C \cup \{x\}) > -h(C)$ but $-g(C \cup \{x\}) = -g(C) = 0$, and hence $\Delta_x p(C) > 0$, a contradiction again. □

Corollary 2.39 *When it is applied to the problem* MIN-WCVC *on connected graphs of at least three vertices, with respect to the potential function* p, Algorithm 2.D *is a polynomial-time* $H(\delta - 1)$*-approximation, where* δ *is the maximum vertex degree of the input graph* G.

Proof. It follows from Theorem 2.29 and the facts that the maximum value of $|E| - g(\{x\})$ is equal to δ and that $-h(\{x\}) = -1$ for all $x \in V$. □

The next example is a 0–1 integer programming problem.

GENERAL COVER (GC): Given nonnegative integers a_{ij}, b_i, and c_j, for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$,

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n} c_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \qquad i = 1, 2, \ldots, m, \\ & x_j \in \{0, 1\}, \qquad j = 1, 2, \ldots, n. \end{aligned}$$

We define a function $f : 2^{\{1,\ldots,n\}} \to \mathbb{N}$ as follows: For any $J \subseteq \{1, \ldots, n\}$,

$$f(J) = \sum_{i=1}^{m} \min \left\{ b_i, \sum_{\ell \in J} a_{i\ell} \right\}.$$

Let $I(J) = \{i \mid \sum_{\ell \in J} a_{i\ell} < b_i\}$. Then it is clear that for any $j, k \in \{1, 2, \ldots, n\}$,

$$\Delta_j f(J) = \sum_{i \in I(J)} \min\left\{a_{ij}, b_i - \sum_{\ell \in J} a_{i\ell}\right\}, \quad \text{and}$$

$$\Delta_j f(J \cup \{k\}) = \sum_{i \in I(J \cup \{k\})} \min\left\{a_{ij}, b_i - \sum_{\ell \in J} a_{i\ell} - a_{ik}\right\}.$$

Moreover, it is not hard to verify that for any $1 \le k \le n$, $I(J \cup \{k\}) \subseteq I(J)$. Thus, $\Delta_j f(J \cup \{k\}) \le \Delta_j f(J)$ for all sets $J \subseteq \{1, 2, \ldots, n\}$ and all $j, k \in \{1, 2, \ldots, n\}$. Thus, by Lemma 2.36, f is a monotone increasing, submodular function.

The collection Ω_f consists of all sets $J \subseteq \{1, 2, \ldots, n\}$ with the maximum value $f(J) = \sum_{i=1}^{n} b_i$. So, Algorithm 2.D and Theorem 2.29 are applicable to problem GC. In particular, the greedy criterion of Algorithm 2.D adds, at each stage, the index j with the maximum value of

$$\frac{1}{c_j} \sum_{i \in I(J)} \min\left\{a_{ij}, b_i - \sum_{\ell \in J} a_{i\ell}\right\}$$

to the solution set J. Also, the parameter γ of the performance ratio $H(\gamma)$ is no more than the maximum value of $\sum_{i=1}^{m} a_{ij}$, $j = 1, 2 \ldots, n$.

Corollary 2.40 *When it is applied to the problem* GC*,* Algorithm 2.D *produces an $H(\gamma)$-approximation in polynomial time, where $\gamma = \max_{1 \le j \le n} \sum_{i=1}^{m} a_{ij}$.*

Finally, we consider a problem about matroids. Recall that a base of a matroid $(E, \mathcal{I})$ is just a maximal independent set. Consider the following problem:

MINIMUM-COST BASE (MIN-CB):

Given a matroid $(E, \mathcal{I})$ and a nonnegative function $c : E \to \mathbb{R}^+$,

$$\begin{array}{ll} \text{minimize} & c(I) \\ \text{subject to} & I \in \mathcal{B}, \end{array}$$

where $\mathcal{B}$ is the family of all bases of the matroid $(E, \mathcal{I})$.

Recall the function *rank* on a matroid $(E, \mathcal{I})$ defined in Example 2.21(b). Then *rank* is a normalized, monotone increasing, submodular function, and it has $\Omega_{rank} = \mathcal{B}$. Therefore, MIN-CB is a special case of MIN-SMC with the potential function $rank$. Note that the corresponding parameter γ in Theorem 2.29 is $\gamma = \max_{x \in E} rank(\{x\}) = 1$, and hence $H(\gamma) = 1$. In other words, the greedy Algorithm 2.D for MIN-CB actually gives the optimal solutions.

Corollary 2.41 *When it is applied to the problem* MIN-CB*, the greedy* Algorithm 2.D *produces a minimum solution in polynomial time.*

2.6 Nonsubmodular Potential Functions

When the associated potential function is not submodular, Theorem 2.29 for the greedy algorithm no longer holds. In such circumstances, how do we analyze the performance of the greedy algorithm? We study this problem in this section.

A *dominating set* of a graph $G = (V, E)$ is a subset $D \subseteq V$ such that every vertex is either in D or adjacent to a vertex in D. A *connected dominating set* C is a dominating set with an additional property that it induces a connected subgraph. The following problem has many applications in wireless communication.

> MINIMUM CONNECTED DOMINATING SET (MIN-CDS): Given a connected graph $G = (V, E)$, find a connected dominating set of G with the minimum cardinality.

Consider a graph G and a subset C of vertices in G. Divide vertices in G into three classes with respect to C, and assign different colors to them: Vertices that belong to C are colored in *black*; vertices that are not in C but are adjacent to C are colored in *gray*; and vertices that are neither in C nor adjacent to C are colored in *white*.

Clearly, C is a connected dominating set if and only if there does not exist a white vertex and the subgraph induced by black vertices is connected. This observation suggests that we use the function $g(C) = p(C) + h(C)$ as the potential function in the greedy algorithm, where $p(C)$ is the number of connected components of the subgraph $G|_C$ induced by C, and $h(C)$ is the number of white vertices. It is clear that C is a connected dominating set if and only if $g(C) = 1$. However, the function g is not really a good candidate for the potential function, because a set C may not be a connected dominating set even if $\Delta_x g(C) = 0$ for all vertices x. Figure 2.9 shows such an example, in which $g(C) = p(C) + h(C) = 2 + 0 = 2 > 1$, but $g(C \cup \{x\}) = g(C)$ for all vertices x. This means that if we apply Algorithm 2.D to MIN-CDS with this potential function g, its output is not necessarily a connected dominating set.

In general, we observe that the graph shown in Figure 2.9 is a typical case resulting from Algorithm 2.D with respect to the potential function g.

Lemma 2.42 *Let $G = (V, E)$ be a connected graph, and $C \subseteq V$. If the subgraph $G|_C$ induced by black vertices is not connected but $\Delta_x g(C) = 0$ for all $x \in V$, then*

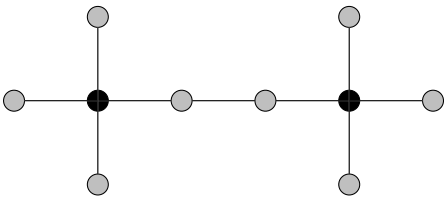


Figure 2.9: $\Delta_x g(C) = 0$ for all vertices x, but C is not a connected dominating set.

all black connected components of $G|_C$ can be connected together through chains of gray vertices, with each chain having exactly two vertices.

Proof. We first note that if $\Delta_x g(C) = 0$ for all $x \in V$, then G has no gray vertex that is adjacent to two black components, since coloring such a gray vertex in black would reduce the value of $g(C)$. In addition, G also has no white verex, for otherwise, by the connectivity of G, there must be a gray vertex adjacent to some white vertex, and coloring this gray vertex in black would reduce the value of $g(C)$, too. Now, suppose, for the sake of contradiction, that some black component cannot be connected to another black component through chains of two adjacent gray vertices. Then, we can divide all black vertices into two parts such that the distance between the two parts is more than 3. Consider a shortest path $\pi = (u, x_1, x_2, \ldots, x_k, v)$ between the two parts, with u and v belonging to the two different parts and $x_1, x_2, \ldots, x_k$ are gray vertices with $k \geq 3$. Since x_2 is gray, it must be adjacent to a black vertex w. If w and u are in the same part, then the path from w to v is a path between the two parts of black vertices shorter than π, which is a contradiction. On the other hand, if w and v are in the same part, then the path from u to w is a path between the two parts shorter than π, also a contradiction. So, the lemma is proven. □

From this lemma, a simple idea of an approximation algorithm works as follows: First, apply the greedy algorithm with the potential function g until $\Delta_x g(C) = 0$ for all $x \in V$. Then, add extra vertices to connect components of $G|_C$. A careful analysis using the pigeonhole principle shows that this modified greedy algorithm achieves the performance ratio $H(\delta) + 3$, where δ is the maximum degree of G (see Section 6.2).

In the following, we take a different approach by choosing a different potential function. Namely, we replace $h(C)$ by $q(C)$, the number of connected components of the subgraph with vertex set V and edge set $D(C)$, where $D(C)$ is the set of all edges incident on some vertices in C. Define $f(C) = p(C) + q(C)$.

Lemma 2.43 *Suppose G is a connected graph with at least three vertices. Then C is a connected dominating set if and only if $f(C \cup \{x\}) = f(C)$ for every $x \in V$.*

Proof. If C is a connected dominating set, then $f(C) = 2$, which reaches the minimum value. Therefore, $f(C \cup \{x\}) = f(C)$ for every $x \in V$.

Conversely, suppose $f(C \cup \{x\}) = f(C)$ for every $x \in V$. First, C cannot be the empty set. In fact, if $C = \emptyset$, then we can pick a vertex x of degree ≥ 2 and get $f(C \cup \{x\}) \leq |V| - 1 < |V| = f(C)$.

So, we may assume $C \neq \emptyset$. Consider a connected component of the subgraph induced by C. Let B denote its vertex set, which is a subset of C, and A be the set of vertices in $V - B$ that are adjacent to a vertex in B. We claim that $V = B \cup A$ (and hence $C = B$ is a connected dominating set for G).

To prove this claim, suppose, by way of contradiction, that $V \neq B \cup A$. Then, since G is connected, there must be a vertex x not in $B \cup A$ that is adjacent to a vertex $y \in B \cup A$. Since all vertices adjacent to B are in A, we know that y must be in A. Now, if x is white or gray, then we must have $p(C \cup \{y\}) \leq p(C)$

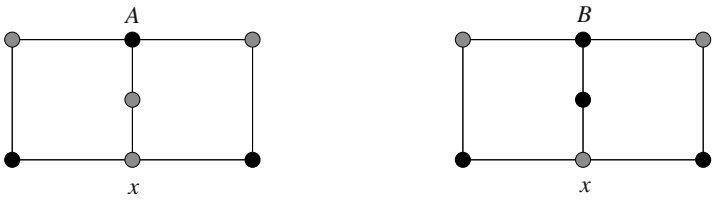


Figure 2.10: A counterexample showing f not supmodular.

and $q(C \cup \{y\}) < q(C)$. If x is black, then we have $p(C \cup \{y\}) < p(C)$ and $q(C \cup \{y\}) \leq q(C)$. In either case, we get $f(C \cup \{y\}) < f(C)$, a contradiction to our assumption. So, the claim, and hence the lemma, is proven. □

This lemma shows that the greedy Algorithm 2.D for MIN-CDS with respect to the potential function f will produce a connected dominating set.

A function $f : 2^E \to \mathbb{R}$ is *supmodular* if $-f$ is submodular. Clearly, all results about monotone increasing, submodular functions can be converted into the results about the corresponding monotone decreasing, supmodular functions. It is easy to see that f is monotone decreasing. Therefore, if f is a supmodular function, then we could directly employ Theorem 2.29 to get the performance ratio of the greedy Algorithm 2.D with respect to f. Unfortunately, as shown in the counterexample of Figure 2.10, f is not supmodular. More specifically, in this example, $A \subseteq B$ but $\Delta_x f(A) = -1 > -2 = \Delta_x f(B)$, and so $-f$ does not satisfy the condition of Lemma 2.36 and is not submodular.

Actually, f is the sum of two functions p and q, where q is supmodular but p is not.

Lemma 2.44 *If $A \subseteq B$, then $\Delta_y q(A) \leq \Delta_y q(B)$.*

Proof. Note that $-\Delta_y q(B) =$ the number of the connected components of the graph $(V, D(B))$ that are adjacent to y but do not contain y. Since each connected component of graph $(V, D(B))$ is constituted by one or more connected components of graph $(V, D(A))$, the number of connected components of $(V, D(B))$ adjacent to y is no more than the number of connected components of $(V, D(A))$ adjacent to y. Thus, we get $-\Delta_y q(B) \leq -\Delta_y q(A)$. □

How do we analyze the performance of the greedy Algorithm 2.D with respect to a nonsubmodular potential function? Let us look at the proof of Theorem 2.22 about the greedy algorithm for MIN-SC again, and see where the submodularity property of the potential function is used. It turns out that it was used only once, when we proved the inequality

$$\Delta_{C_j} f(\mathcal{A}_i) \geq \Delta_{C_j} f(\mathcal{A}_i \cup \mathcal{C}_{j-1}) \tag{2.6}$$

to get (2.4). An important observation about this inequality is that the incremental variables C_j, $1 \leq j \leq m$, are sets of the optimal solution, arranged in an arbitrary order. Therefore, although for nonsubmodular functions f this inequality may not

hold for an arbitrary ordering of sets in the optimal solution, a carefully arranged ordering on these sets might still satisfy, or almost satisfy, this inequality. In the following, we will implement this idea for the problem MIN-CDS.

Let the vertices $x_1, \ldots, x_g$ be the elements of the solution found by Algorithm 2.D with respect to the potential function f, in the order of their selection into the solution set. Denote $C_i = \{x_1, x_2, \ldots, x_i\}$ and consider $f(C_i)$. Initially, $f(C_0) = n$, where n is the number of vertices in G. Let C^* be a minimum connected dominating set for G. Assume that $|C^*| = m$.

Lemma 2.45 *For $i = 1, 2, \ldots, g$,*

$$f(C_i) \leq f(C_{i-1}) - \frac{f(C_{i-1}) - 2}{m} + 1. \tag{2.7}$$

Proof. First, consider the case of $i \geq 2$. We note that

$$f(C_i) = f(C_{i-1}) + \Delta_{x_i} f(C_{i-1}).$$

Since C^* is a connected dominating set, we can always arrange the elements of C^* in an ordering $y_1, y_2, \ldots, y_m$ such that y_1 is adjacent to a vertex in C_{i-1} and, for each $j \geq 2$, y_j is adjacent to a vertex in $\{y_1, \ldots, y_{j-1}\}$. Denote $C^*_j = \{y_1, y_2, \ldots, y_j\}$. Then

$$\Delta_{C^*} f(C_{i-1}) = \sum_{j=1}^{m} \Delta_{y_j} f(C_{i-1} \cup C^*_{j-1}).$$

For each $1 \leq j \leq m$, we note that y_j can dominate at most one additional connected component in the subgraph $G|_{C_{i-1} \cup C^*_{j-1}}$ than in $G|_{C_{i-1}}$, which is the one that contains C^*_{j-1}, since all vertices $y_1, \ldots, y_{j-1}$ in C^*_{j-1} are connected. Since $-\Delta_y p(C)$ is equal to the number of connected components of $G|_C$ that are adjacent to y *minus* 1, it follows that

$$-\Delta_{y_j} p(C_{i-1} \cup C^*_{j-1}) \leq -\Delta_{y_j} p(C_{i-1}) + 1.$$

Moreover, by Lemma 2.44,

$$-\Delta_{y_j} q(C_{i-1} \cup C^*_{j-1}) \leq -\Delta_{y_j} q(C_{i-1}).$$

So we have

$$-\Delta_{y_j} f(C_{i-1} \cup C^*_{j-1}) \leq -\Delta_{y_j} f(C_{i-1}) + 1.$$

[Note that this inequality is close to our desired inequality (2.6).] From this inequality, we get

$$\begin{aligned} f(C_{i-1}) - 2 &= -\Delta_{C^*} f(C_{i-1}) \\ &= \sum_{j=1}^{m} (-\Delta_{y_j} f(C_{i-1} \cup C^*_{j-1})) \leq \sum_{j=1}^{m} (-\Delta_{y_j} f(C_{i-1}) + 1). \end{aligned}$$

By the pigeonhole principle, there exists an element $y_j \in C^*$ such that

$$-\Delta_{y_j} f(C_{i-1}) + 1 \geq \frac{f(C_{i-1}) - 2}{m}.$$

By the greedy strategy of Algorithm 2.D,

$$-\Delta_{x_i} f(C_{i-1}) \geq -\Delta_{y_j} f(C_{i-1}) \geq \frac{f(C_{i-1}) - 2}{m} - 1.$$

Or, equivalently,

$$f(C_i) \leq f(C_{i-1}) - \frac{f(C_{i-1}) - 2}{m} + 1.$$

For the case of $i = 1$, the proof is essentially identical, with the difference that y_1 could be an arbitrary vertex in C^*. □

Theorem 2.46 *When it is applied to the problem* MIN-CDS *with respect to the potential function $-f$, the greedy* Algorithm 2.D *is a polynomial-time $(2 + \ln \delta)$-approximation, where δ is the maximum degree of the input graph.*

Proof. If $g \leq 2m$, then the proof is already done. So we assume that $g > 2m$.

Rewrite the inequality (2.7) as

$$f(C_i) - 2 \leq (f(C_{i-1}) - 2)\Big(1 - \frac{1}{m}\Big) + 1.$$

Solving this recurrence relation, we have

$$\begin{aligned} f(C_i) - 2 &\leq (f(C_0) - 2)\Big(1 - \frac{1}{m}\Big)^i + \sum_{k=0}^{i-1} \Big(1 - \frac{1}{m}\Big)^k \\ &= (f(C_0) - 2)\Big(1 - \frac{1}{m}\Big)^i + m\Big(1 - \Big(1 - \frac{1}{m}\Big)^i\Big) \\ &= (f(C_0) - 2 - m)\Big(1 - \frac{1}{m}\Big)^i + m. \end{aligned}$$

From the greedy strategy of Algorithm 2.D, we reduce the value $f(C_{i-1})$ in each stage $i \leq g$. Therefore, $f(C_i) \leq f(C_{i-1}) - 1$. In addition, $f(C_g) = 2$. So we have $f(C_{g-2m}) \geq 2m + 2$. Set $i = g - 2m$, and observe that

$$2m \leq f(C_i) - 2 \leq (n - 2 - m)\Big(1 - \frac{1}{m}\Big)^i + m,$$

where n is the number of vertices in G. Since $(1 - 1/m)^i \leq e^{-i/m}$, we obtain

$$i \leq m \cdot \ln \frac{n - 2 - m}{m}.$$

Note that each vertex has at most δ neighbors and so can dominate at most $\delta + 1$ vertices. Hence, $n/m \leq \delta + 1$. It follows that $g = i + 2m \leq m(2 + \ln \delta)$. □

Now, let us consider another simple idea for designing greedy algorithms with respect to a nonsubmodular potential function. In the greedy Algorithm 2.C for the problem MIN-SC, we add, in each iteration, one subset C to the solution $\mathcal{A}$. Suppose we are allowed to add two or more subsets to $\mathcal{A}$ in each iteration. Does this give us a better performance ratio? It is easy to see that the answer is no. In general, does this idea work for the greedy Algorithm 2.D with respect to a submodular potential function f? The answer is again no, since a submodular function satisfies the property of Lemma 2.23. On the other hand, if the potential function f is not submodular, then this idea may actually work. In the following, we show that the greedy algorithm based on this idea actually gives a better performance ratio for MIN-CDS than Algorithm 2.D. More precisely, the performance ratio of the following greedy algorithm for MIN-CDS approaches $1 + \ln \delta$, as k tends to ∞.

Algorithm 2.E (*Greedy Algorithm for* MIN-CDS)
Input: A connected graph $G = (V, E)$ and an integer $k \geq 2$.

(1) $C \leftarrow \emptyset$.

(2) **While** $f(C) > 2$ **do**

Select a set $X \subseteq V$ of size $|X| \leq 2k - 1$ that maximizes $\dfrac{-\Delta_X f(C)}{|X|}$;

Set $C \leftarrow C \cup X$.

(3) Output $C_g \leftarrow C$. ■

To analyze greedy Algorithm 2.E, we note the following property of the potential function $-f$.

Lemma 2.47 *Let A, B, and X be three vertex subsets. If both $G|_B$ and $G|_X$ are connected, then*

$$-\Delta_X f(A \cup B) + \Delta_X f(A) \leq 1.$$

Proof. Since q is supmodular, we have $\Delta_X q(A) \leq \Delta_X q(A \cup B)$.

For function p, we note that, since $G|_X$ is connected, $-\Delta_X p(A)$ is equal to the number of black components dominated by X in graph $G|_A$ minus 1. Since the subgraph $G|_B$ is connected, the number of black components dominated by X in $G|_{A \cup B}$ is at most one more than the number of black components dominated by X in $G|_A$. Therefore, we have $-\Delta_X p(A \cup B) \leq -\Delta_X p(A) + 1$. It follows that $-\Delta_X f(A \cup B) \leq -\Delta_X f(A) + 1$. □

Let C^* be a minimum solution to MIN-CDS. We show two properties of C^* in the following two lemmas.

Lemma 2.48 *For any integer $k \geq 2$, C^* can be decomposed into $Y_1, Y_2, \ldots, Y_h$, for some $h \geq 1$, such that*

(a) $C^* = Y_1 \cup Y_2 \cup \cdots \cup Y_h$;

(b) For each $1 \leq i \leq h$, both $G|_{Y_1 \cup Y_2 \cup \cdots \cup Y_i}$ and $G|_{Y_i}$ are connected;

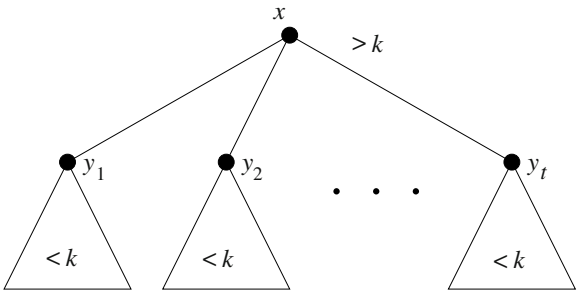


Figure 2.11: Case 2 in proof of Lemma 2.48.

(c) *For each* $1 \le i \le h$, $1 \le |Y_i| \le 2k-1$; *and for all but one* $1 \le i \le h$, $k+1 \le |Y_h|$; *and*

(d) $|Y_1| + |Y_2| + \cdots + |Y_h| \le |C^*| + h - 1$.

Proof. We can construct sets $Y_1, \ldots, Y_h$ recursively.

Let T be a subtree of $G|_{C^*}$ that contains all vertices in C^*. Choose an arbitrary vertex $r \in C^*$ as the root of T. For any vertex $x \in C^*$, let $T(x)$ denote the subtree of T rooted at x, and $|T(x)|$ the number of vertices in $T(x)$.

If $|T| \le 2k-1$, then let $Y_1 = C^*$ and the lemma holds with $h = 1$. If T contains more than $2k-1$ vertices, then there must exist a vertex $x \in C^*$ such that $|T(x)| \ge k+1$ and for every child y of x, $|T(y)| \le k$. Now, consider two cases.

Case 1. There is a child y of x such that $|T(y)| = k$. Let Y_1 consist of all vertices of $T(y)$ together with x and delete all vertices of $T(y)$ from T.

Case 2. For every child y of x, $|T(y)| \le k-1$. Suppose $y_1, \ldots, y_t$ are all children of x (cf. Figure 2.11). There must exist an integer $1 \le j \le t-1$ such that

$$|T(y_1)| + \cdots + |T(y_j)| \le k-1$$

and

$$|T(y_1)| + \cdots + |T(y_j)| + |T(y_{j+1})| \ge k.$$

Since $|T(y_{j+1})| \le k-1$, we have

$$|T(y_1)| + \cdots + |T(y_j)| + |T(y_{j+1})| \le 2k-2.$$

Let Y_1 consist of all vertices in $T(y_1) \cup \cdots \cup T(y_{j+1})$ together with x and delete $Y_1 - \{x\}$ from T.

Repeating the above process on the remaining T, and rearranging the order of the sets $Y_1, \ldots, Y_h$, we will obtain a required decomposition. □

Lemma 2.49 *Let* δ *be the maximum degree of* $G = (V, E)$. *Then we have* $|V| \le (\delta - 1)|C^*| + 2$.

Proof. We prove by induction on $|C|$ that a subset C of V with connected $G|_C$ can dominate at most $(\delta-1)|C|+2$ vertices. For $|C| = 1$, it is trivially true. For $|C| \geq 2$, choose a vertex $x \in C$ such that $G|_{C-\{x\}}$ is still connected. Since x has at most δ neighbors, and at least one of them is in $C - \{x\}$, we see that C dominates at most $\delta - 1$ more vertices than $C - \{x\}$ does. By the induction hypothesis, $C - \{x\}$ can dominate at most $(\delta - 1)(|C| - 1) + 2$ vertices. Therefore, C can dominate at most $(\delta - 1)|C| + 2$ vertices. □

Theorem 2.50 *For any $\varepsilon > 0$, there exists a polynomial-time approximation with performance ratio $(1 + \varepsilon)\ln(\delta - 1)$ for* MIN-CDS*, where δ is the maximum degree of the input graph.*

Proof. Let $G = (V, E)$ be a connected graph with the maximum degree δ. We can find easily a minimum connected dominating set of G if $\delta \leq 2$: If $\delta = 1$, then G contains only one edge, and either vertex of the edge is a minimum connected dominating set. If $\delta = 2$, G is either a path or a cycle, and a minimum connected dominating set of G can be obtained by deleting, respectively, either the two leaves or any two adjacent vertices.

For graphs with $\delta \geq 3$, we consider Algorithm 2.E on G. Let $X_1, \ldots, X_g$ be the sets chosen by greedy Algorithm 2.E on graph G, in the order of their selection into set C. Denote $C_i = X_1 \cup \cdots \cup X_i$, for $0 \leq i \leq g$ (in particular, C_g is the output of Algorithm 2.E). Let C^* be a minimum connected dominating set for G, and $m = |C^*|$. Decompose C^* into $Y_1, Y_2, \ldots, Y_h$, satisfying conditions given in Lemma 2.48. Denote $C^*_j = Y_1 \cup \cdots \cup Y_j$, for $0 \leq j \leq h$.

From Lemma 2.48, we know that $G|_{Y_j}$ and $G|_{C^*_j}$ are connected for each $1 \leq j \leq h$. Thus, we have, by Lemma 2.47,

$$-\Delta_{Y_j} f(C_i \cup C^*_{j-1}) \leq -\Delta_{Y_j} f(C_i) + 1,$$

for $0 \leq i \leq g$ and $1 \leq j \leq h$. By the greedy rule of Algorithm 2.E, we get

$$\frac{-\Delta_{X_{i+1}} f(C_i)}{|X_{i+1}|} \geq \frac{-\Delta_{Y_j} f(C_i)}{|Y_j|},$$

for $0 \leq i \leq g$ and $1 \leq j \leq h$. Note that $f(C^*) = 2$ and, hence, for $0 \leq i \leq g - 1$,

$$\begin{aligned}
\frac{-\Delta_{X_{i+1}} f(C_i)}{|X_{i+1}|} &\geq \frac{-\sum_{j=1}^{h} \Delta_{Y_j} f(C_i)}{\sum_{j=1}^{h} |Y_j|} \\
&\geq \frac{-(h-1) - \sum_{j=1}^{h} \Delta_{Y_j} f(C_i \cup C^*_{j-1})}{\sum_{j=1}^{h} |Y_j|} \\
&\geq \frac{-(h-1) - (f(C_i \cup C^*) - f(C_i))}{m + h - 1} \\
&= \frac{f(C_i) - (h+1)}{m + h - 1}.
\end{aligned}$$

Denote $a_i = f(C_i) - (h+1)$. Then the above inequality can be rewritten as

$$\frac{a_i - a_{i+1}}{|X_{i+1}|} \;\geq\; \frac{a_i}{m+h-1}, \quad \text{for } 0 \leq i \leq g-1.$$

That is, for each $0 \leq i \leq g-1$,

$$\begin{aligned} a_{i+1} \;&\leq\; a_i\Big(1 - \frac{|X_{i+1}|}{m+h-1}\Big) \leq a_i \cdot \exp\Big(\frac{-|X_{i+1}|}{m+h-1}\Big) \\ &\leq\; a_0 \cdot \exp\Big(\frac{-(|X_{i+1}| + |X_i| + \cdots + |X_1|)}{m+h-1}\Big). \end{aligned} \tag{2.8}$$

Fix the index i, $0 \leq i \leq g-1$, such that

$$a_i \geq m > a_{i+1},$$

and let $b = a_i - m$ and $b' = m - a_{i+1}$. Write $|X_{i+1}| = d + d'$ such that

$$\frac{b}{d} = \frac{b'}{d'} = \frac{a_i - a_{i+1}}{|X_{i+1}|} \geq \frac{a_i}{m+h-1}.$$

(In case of $b = 0$, just let $d' = |X_{i+1}|$.) We now divide the greedy solution $|C_g|$ into two parts, $|X_1| + \cdots + |X_i| + d$, and $d' + |X_{i+2}| + \cdots + |X_g|$, and bound them separately.

For the first part, we note that

$$\frac{a_i - m}{d} = \frac{b}{d} \geq \frac{a_i}{m+h-1},$$

and so

$$m \leq a_i\Big(1 - \frac{d}{m+h-1}\Big) \leq a_i \cdot e^{-d/(m+h-1)}.$$

Combining this with (2.8), we get

$$m \leq a_0 \cdot e^{-(d+|X_i|+\cdots+|X_1|)/(m+h-1)}.$$

Note that $a_0 = f(\emptyset) - (h+1) = |V| - (h+1)$. Thus,

$$|X_1| + \cdots + |X_i| + d \leq (m+h-1)\ln\frac{|V| - (h+1)}{m}.$$

For the second part, we note that $-\Delta_{X_{j+1}} f(C_j)/|X_{j+1}| \geq 1$ for all $0 \leq j \leq g-1$, since we can, by Lemma 2.43, always find a vertex v to make $-\Delta_{\{v\}} f(C_j) \geq 1$. That is,

$$|X_{j+1}| \leq f(C_j) - f(C_{j+1}),$$

for $0 \leq j \leq g-1$. Thus,

$$\begin{aligned} d' + |X_{i+2}| + \cdots + |X_g| &\leq b' + f(C_{i+1}) - f(C_g) \\ &= m - a_{i+1} + f(C_{i+1}) - f(C^*) = m + h - 1. \end{aligned}$$

Together, we have

$$|X_1| + \cdots + |X_g| \le (m + h - 1)\Big(1 + \ln \frac{|V| - (h+1)}{m}\Big).$$

From conditions (c) and (d) of Lemma 2.48, we know that

$$(h-1)(k+1) + 1 \le |Y_1| + |Y_2| + \cdots + |Y_h| \le m + h - 1,$$

and hence

$$h - 1 \le \frac{m}{k}.$$

Moreover, by Lemma 2.49, $|V| \le (\delta - 1)m + 2$. Since $h \ge 1$, we have

$$\frac{|V| - (h+1)}{m} \le \delta - 1.$$

Therefore,

$$|X_1| + \cdots + |X_g| \le m\Big(1 + \frac{1}{k}\Big)\Big(1 + \ln(\delta - 1)\Big).$$

Now, the theorem follows by choosing k such that $1/k < \varepsilon$. □

Exercises

2.1 Let $(E, \mathcal{I})$ be an independent system. Suppose that all maximal independent subsets of E have cardinality k. Define

$$p = \max_{F \subseteq E} \frac{v(F)}{u(F)},$$

where $u(F)$ and $v(F)$ are the functions defined in (2.1). Let $c : E \to \mathbb{R}^+$ be a nonnegative cost function on E. Also, let I^* be a maximal independent subset of E with the minimum cost, and I_G an independent subset obtained by greedy Algorithm 2.A on the problem MAX-ISS. Prove that

$$c(I^*) \le c(I_G) \le \frac{1}{p} \cdot c(I^*) + \frac{p-1}{p} \cdot k \cdot M,$$

where $M = \max_{e \in E} c(e)$.

2.2 For a complete directed graph $G = (V, E)$, let $\mathcal{I}_G$ be the family of the edge sets of all acyclic subgraphs of G. Show that for any integer $k > 0$, there exists a complete directed graph $G = (V, E)$ such that for the independent system $(E, \mathcal{I}_G)$,

$$\max_{F \subseteq E} \frac{v(F)}{u(F)} \ge k.$$

2.3 Show that for every integer $k \geq 1$, there exists an independent system $(E, \mathcal{I})$ that is an intersection of k matroids but not an intersection of less than k matroids, such that

$$\max_{F \subseteq E} \frac{v(F)}{u(F)} = k.$$

2.4 Prove that an independent system $(E, \mathcal{I})$ is a matroid if and only if, for any cost function $c : E \to \mathbb{N}^+$, the greedy Algorithm 2.D produces a minimum solution for MIN-CB.

2.5 Prove that the distance function defined in the transformation from the problem SS to the problem TSP, as described at the end of Section 2.3, satisfies the triangle inequality.

2.6 Prove that for every positive integer m, $\sum_{i=1}^{m} 1/i \leq 1 + \ln m$.

2.7 In terms of the notion of hypergraphs, the problem MIN-SC asks for a minimum-size hyperedge set that is incident on each vertex of the input hypergraph. A k-matching in a hypergraph H is a sub-hypergraph of degree at most k. Let m_k be the maximum number of edges in a k-matching. Prove that

(a) $m_k \leq k \cdot |C^*|$, where C^* is a minimum set cover of H, and

(b) $|C_G| \leq \sum_{i=1}^{d} m_i/(i(i+1)) + m_d/d$, where C_G is the output of the greedy Algorithm 2.C, and d is the maximum degree of H.

2.8 Use Exercise 2.7 to give another proof to Theorem 2.22.

2.9 Let $G = (V, E)$ be a graph and $c : E \to 2^{\mathbb{N}}$ a color-set function (i.e., $c(e)$ is a *color set* for edge e). A *color-covering* of the graph G is a color set $C \subseteq \mathbb{N}$ such that the set of edges e with $c(e) \cap C \neq \emptyset$ contains a spanning tree of G. Prove that the following problem has a polynomial-time $(1 + \ln |V|)$-approximation:

> For a given graph G and a given color-set function $c : E \to 2^N$, find a color-covering of the minimum cardinality.

2.10 Show that the following problem has a polynomial-time $(2 + \ln |V|)$-approximation:

> Given a graph $G = (V, E)$ and a color-set function $c : E \to 2^{\mathbb{N}}$, find the subset $C \subseteq V$ of the minimum cardinality such that all colors of the edges incident upon the vertices in C form a color-covering of G.

2.11 A function $g : \mathbb{N} \to \mathbb{R}^+$ is a *concave function* if, for any $m, r, n \in \mathbb{N}$, with $m < r < n$, $g(r) \geq tg(m) + (1-t)g(n)$, where $t = (n-r)/(n-m)$. Let E be a finite set, and let f be a real function defined on 2^E such that $f(A) = g(|A|)$ for all $A \subseteq E$. Show that f is submodular if and only if g is concave.

2.12 Consider a graph $G = (V, E)$. Let $\bar{\delta}(X)$ for $X \subseteq V$ denote the set of edges between X and $V - X$. Show that $|\bar{\delta}(X)|$ is a submodular function.

2.13 Show that a function f on 2^E is modular (both submodular and supmodular) if and only if f is linear.

2.14 Prove Lemmas 2.35 and 2.36.

2.15 Suppose f and c are two polymatroid functions on 2^E, and f is an integer function. Consider the problem MIN-SMC with a possibly nonlinear cost function c; i.e., the problem of minimizing $c(A)$ over $\{A \subseteq E \mid f(A) = f(E)\}$. Show that the greedy Algorithm 2.D for MIN-SMC is a $(\rho \cdot H(\gamma))$-approximation, where $\gamma = \max\{f(\{x\}) \mid x \in E\}$ and ρ is the *curvature* of c, defined by

$$\rho = \min\left\{ \frac{\sum_{e \in S} c(e)}{c(S)} \;\middle|\; f(S) = f(E) \right\}.$$

2.16 Consider a digraph $G = (V, E)$. For $X \subseteq V$, let $\bar{\delta}_+(X)$ ($\bar{\delta}_-(X)$) denote the set of edges going out from (coming into, respectively) X. Show that $|\bar{\delta}_+(X)|$ and $|\bar{\delta}_-(X)|$ are submodular functions.

2.17 Let r be a function mapping 2^E to $\mathbb{N}$. Show that the following statements are equivalent:

(a) $\mathcal{I} = \{I \subseteq E \mid r(I) = |I|\}$ defines a matroid $(E, \mathcal{I})$ and r is its rank function.

(b) For all $A, B \subseteq E$, r satisfies the following conditions:

(i) $r(A) \leq |A|$;

(ii) if $A \subseteq B$, then $r(A) \leq r(B)$; and

(iii) r is submodular.

2.18 Show that a polymatroid (E, r) is a matroid if and only if $r(\{x\}) = 1$ for every $x \in E$.

2.19 Suppose $(E, r_1), (E, r_2), \ldots, (E, r_k)$ are matroids. Show that $(E, \sum_{i=1}^{k} r_i)$ is a polymatroid.

2.20 Let $(E, \mathcal{I})$ be a matroid, and *rank* its rank function. Consider a collection $\mathcal{C}$ of subsets of E. For $\mathcal{A} \subseteq \mathcal{C}$, define

$$f(\mathcal{A}) = \mathit{rank}\big(\textstyle\bigcup_{A \in \mathcal{A}} A\big).$$

Show that (E, f) is a polymatroid.

2.21 Show that for any polymatroid $(\mathcal{E}, f)$, there exist a matroid (E, r) and a one-to-one mapping $\phi : \mathcal{E} \to 2^E$ such that

$$f(\mathcal{A}) = r\big(\textstyle\bigcup_{A \in \phi(\mathcal{A})} A\big).$$

2.22 For any polymatroid (E, f), define f^d on 2^E with

$$f^d(S) = \sum_{j \in S} f(\{j\}) - f(E) - f(E - S).$$

Show that (E, f^d) is still a polymatroid. [It is called the *dual polymatroid* of (E, f).]

2.23 For any polymatroid (E, f), let $\mathcal{I} = \{A \mid f(A) = |A|, A \subseteq E\}$. Show that $(E, \mathcal{I})$ is an independent system.

2.24 Let $(E, \mathcal{I})$ be an independent system. Define $r(A) = \max\{|I| \mid I \in \mathcal{I}, I \subseteq A\}$. Give an example of $(E, \mathcal{I})$ for which r is not a polymatroid function.

2.25 Let (E, f) be a polymatroid and c a nonnegative cost function on E. Show that the problem of computing $\min\{c(A) \mid f(A) \geq k, A \subseteq E\}$ has a greedy approximation with performance ratio $H(\min\{k, \gamma\})$, where $\gamma = \max_{x \in E} f(\{x\})$.

2.26 Consider the application of Algorithm 2.D to MIN-CDS with the potential function $f(C) = p(C) + q(C)$. Find a graph G on which the algorithm produces an approximate solution of size $g \leq 2|C^*|$.

2.27 Given a hypergraph $H = (V, S)$ and a function $f : S \to \mathbb{N}^+$, find a minimum vertex cover C such that for every hyperedge $s \in S$, $|C \cap s| \geq f(s)$. Prove that this problem has a polynomial-time $(1 + \ln d)$-approximation, where d is the maximum vertex degree in H.

2.28 Let $f : 2^E \to \mathbb{R}$ be a normalized submodular function. We associate a weight $w_i \geq 0$ with each $i \in E$. Consider the following linear program:

$$\begin{aligned} \text{maximize} \quad & \sum_{i \in E} w_i x_i \\ \text{subject to} \quad & \sum_{i \in A} x_i \leq f(A), \quad A \subseteq E. \end{aligned}$$

Show that this problem can be solved by the following greedy algorithm:

(1) Sort elements of E and rename them so that $w_1 \geq w_2 \geq \cdots \geq w_n$.

(2) $A_0 \leftarrow \emptyset$; **for** $k \leftarrow 1$ **to** n **do** $A_k \leftarrow \{1, 2, \ldots, k\}$.

(3) **For** $k \leftarrow 1$ **to** n **do** $x_i \leftarrow f(A_i) - f(A_{i-1})$.

2.29 Let E be a finite set and $p : E \to \mathbb{R}^+$ a positive function on E. For every subset A of E, define

$$g(A) = \left(\sum_{i \in A} p(i) \right)^2 + \sum_{i \in A} p(i)^2.$$

Show that g is a supmodular function.

2.30 Show that the following greedy algorithm for the problem MIN-CDS has performance ratio $2(1 + H(\delta))$, where δ is the maximum vertex degree:

> Grow a tree T starting from a vertex of the maximum degree. At each iteration, add one or two adjacent vertices to maximize the increase in the number of dominated vertices.

2.31 In the proof of Lemma 2.45, a simple argument has been suggested as follows:

> Since $m = |C^*|$ vertices are able to reduce the total number of connected components in the two subgraphs from $f(C_{i-1})$ to 2, there must exist a vertex that is able to reduce at least $\lceil (f(C_{i-1}) - 2)/m \rceil - 1$ components (here, the term -1 comes from considering the increase in the number of black components). Therefore, $-\Delta_{x_i} f(C_{i-1}) \geq (f(C_{i-1}) - 2)/m - 1$, and hence the lemma holds.

Find the error of this argument and explain why with a counterexample to the above statement.

2.32 Give a counterexample to show that Lemma 2.47 does not hold if $G|_X$ is not connected.

2.33 A dominating set A in a graph is said to be *weakly connected* if all edges incident upon vertices in A induce a connected subgraph. Show that there exists a greedy $H(\delta)$-approximation for the problem of finding the minimum-size weakly connected dominating set of a given graph, where δ is the maximum vertex degree of the input graph.

2.34 Consider a hypergraph $(V, \mathcal{E})$, where $\mathcal{E}$ is a collection of subsets of V. A subcollection $\mathcal{C}$ of $\mathcal{E}$ is called a *connected set cover* if $\mathcal{C}$ is a set cover of V and $(V, \mathcal{C})$ is a connected sub-hypergraph. Show that the problem of finding a connected set cover with the minimum cardinality has a greedy $H(\delta)$-approximation, where δ is the maximum vertex degree of the input hypergraph.

2.35 Consider a hypergraph $(V, \mathcal{E})$, where $\mathcal{E}$ is a collection of subsets of V. A subset A of V is called a *dominating set*, if every vertex is either in A or adjacent to A. Furthermore, A is said to be *connected* if A induces a connected sub-hypergraph. Design a greedy approximation for computing the minimum connected dominating set in hypergraphs. Could you reach approximation ratio $(1 - \varepsilon)(1 + \ln \delta)$ for any $\varepsilon > 0$, where δ is the maximum vertex degree of the input hypergraph?

2.36 A set S of sensors is associated with a graph $G = (S, E)$, and each sensor $s \in S$ can monitor a set T_s of targets. Let T be the collection of all targets; i.e., $T = \bigcup_{s \in S} T_s$. Consider the following problem:

> CONNECTED TARGET COVERAGE (CTC): Given a sensor graph $G = (S, E)$ and, for each sensor $s \in S$, a target set T_s, find a minimum-cardinality subset A of S such that A can monitor all targets in T and such that A also induces a connected subgraph of G.

Design a greedy approximation for CTC and analyze the performance ratio of your algorithm.

Historical Notes

The analysis of the greedy algorithm for independent systems was first reported by Jenkyns [1976] and Korte and Hausmann [1978]. Hausmann, Korte, and Jenkyns [1980] further studied algorithms of this type. Submodular set functions play an important role in combinatorial optimization. Some of the results presented in Section 2.4 can be found in Wolsey [1982a].

Lund and Yannakakis [1994] proved that for any $0 < \rho < 1/4$, there is no polynomial-time approximation algorithm with performance ratio $\rho \ln n$ for MIN-SC unless **NP** $\subseteq$ **DTIME**$(n^{poly \log n})$. Feige [1998] improved this result by relaxing ρ to $0 < \rho < 1$. This means that it is unlikely for MIN-SC to have a constant-bounded polynomial-time approximation. Johnson [1974] and Lovász [1975] independently discovered a polynomial-time greedy $H(\delta)$-approximation for MIN-SC. Chvátal [1979] extended the greedy approximation to the weighted case. The greedy algorithm for MIN-SC can be analyzed in many ways. Slavik [1997] presented a tight one. The problem WSID was proposed by Du and Miller [1988]. Prisner [1992] presented a greedy approximation for it and claimed that it has performance ratio $1 + \ln K$. Unfortunately, his proof contained an error. Du, Wu, and Kelley [1998] fixed this error. They also showed, based on a reduction from the problem MIN-SC, a lower bound on the performance ratio for WSID. It is known that the problem MIN-CDS is **NP**-hard [Garey and Johnson, 1978]. Guha and Khuller [1998a] presented a greedy algorithm for it with performance ratio $3 + \ln \delta$. Ruan et al. [2003] gave a new one with performance $2 + \ln \delta$. The $(1 + \varepsilon)(1 + \ln \delta)$-approximation can be found in Du et al. [2008].

http://www.springer.com/978-1-4614-1700-2

Design and Analysis of Approximation Algorithms

Du, D.-Z.; Ko, K.-I.; Hu, X.

2012, XII, 440 p., Hardcover
ISBN: 978-1-4614-1700-2