Chapter 45

Submodular function minimization

This chapter describes a strongly polynomial-time algorithm to find the minimum value of a submodular function. It suffices that the submodular function is given by a value giving oracle.
One application of submodular function minimization is optimizing over the intersection of two polymatroids. This will be discussed in Chapter 47.

45.1. Submodular function minimization

It was shown by Grötschel, Lovász, and Schrijver [1981] that the minimum value of a rational-valued submodular set function f on S can be found in polynomial time, if f is given by a value giving oracle and an upper bound B is given on the numerators and denominators of the values of f. The running time is bounded by a polynomial in $|S|$ and $\log B$. This algorithm is based on the ellipsoid method: we can assume that $f(\emptyset) = 0$ (by resetting $f(U) := f(U) - f(\emptyset)$ for all $U \subseteq S$); then with the greedy algorithm, we can optimize over EP_f in polynomial time (Corollary 44.3b), hence the separation problem for EP_f is solvable in polynomial time, hence also the separation problem for

(45.1) $\qquad P := EP_f \cap \{x \mid x \leq \mathbf{0}\},$

and therefore also the optimization problem for P. Now the maximum value of $x(S)$ over P is equal to the minimum value of f (by (44.8), (44.9), and (44.34)).

Having a polynomial-time method to find the minimum value of a submodular function, we can turn it into a polynomial-time method to find a subset T of S minimizing $f(T)$: For each $s \in S$, we can determine if the minimum value of f over all subsets of S is equal to the minimum value of f over subsets of $S \setminus \{s\}$. If so, we reset $S := S \setminus \{s\}$. Doing this for all elements of S, we are left with a set T minimizing f over all subsets of (the original) S.

Grötschel, Lovász, and Schrijver [1988] showed that this algorithm can be turned into a strongly polynomial-time method. Cunningham [1985b] gave a combinatorial, pseudo-polynomial-time algorithm for minimizing a submodular function f (polynomial in the size of the underlying set and the maximum absolute value of f (assuming f to be integer)). Inspired by Cunningham's method, combinatorial strongly polynomial-time algorithms were found by Iwata, Fleischer, and Fujishige [2000,2001] and Schrijver [2000a]. We will describe the latter algorithm.

45.2. Orders and base vectors

Let f be a submodular set function on a set S. In finding the minimum value of f, we can assume $f(\emptyset) = 0$, as resetting $f(U) := f(U) - f(\emptyset)$ for all $U \subseteq S$ does not change the problem. So throughout we assume that $f(\emptyset) = 0$.

Moreover, we assume that f is given by a *value giving oracle*, that is, an oracle that returns $f(U)$ for any given subset U of S. We also assume that the numbers returned by the oracle are rational (or belong to any ordered field in which we can perform the elementary arithmetic operations algorithmically).

Recall that the *base polytope* B_f of f is defined as the set of base vectors of f:

(45.2) $$B_f := \{x \in \mathbb{R}^S \mid x(U) \leq f(U) \text{ for all } U \subseteq S, x(S) = f(S)\}.$$

Consider any total order $\prec$ on S.[35] For any $v \in S$, denote

(45.3) $$v_\prec := \{u \in S \mid u \prec v\}.$$

Define a vector $b^\prec$ in $\mathbb{R}^S$ by:

(45.4) $$b^\prec(v) := f(v_\prec \cup \{v\}) - f(v_\prec)$$

for $v \in S$. Theorem 44.3 implies that $b^\prec$ belongs to B_f.

Note that $b^\prec(U) = f(U)$ for each lower ideal U of $\prec$ (where a *lower ideal* of $\prec$ is a subset U of S such that if $v \in U$ and $u \prec v$, then $u \in U$).

45.3. A subroutine

In this section we describe a subroutine that is important in the algorithm. It replaces a total order $\prec$ by other total orders, thereby reducing some interval $(s,t]_\prec$, where

(45.5) $$(s,t]_\prec := \{v \mid s \prec v \preceq t\}$$

for $s, t \in S$.

Let $\prec$ be a total order on S. For any $s, u \in S$ with $s \prec u$, let $\prec^{s,u}$ be the total order on S obtained from $\prec$ by resetting $v \prec u$ to $u \prec v$ for each

[35] As usual, we use $\prec$ for strict inequality and $\preceq$ for nonstrict inequality. We refer to the order by the strict inequality sign $\prec$.

v satisfying $s \preceq v \prec u$. Thus in the ordering, we move u to the position just before s. Hence $(s,t]_{\prec^{s,u}} = (s,t]_{\prec} \setminus \{u\}$ if $u \in (s,t]_{\prec}$.

We show that there is a strongly polynomial-time subroutine that

(45.6) for any $s,t \in S$ with $s \prec t$, finds a $\delta \geq 0$ and describes $b^{\prec} + \delta(\chi^t - \chi^s)$ as a convex combination of the $b^{\prec^{s,u}}$ for $u \in (s,t]_{\prec}$.

To describe the subroutine, we can assume that $b^{\prec} = \mathbf{0}$, by replacing (temporarily) $f(U)$ by $f(U) - b^{\prec}(U)$ for each $U \subseteq S$.

We investigate the signs of the vector $b^{\prec^{s,u}}$. We show that for each $v \in S$:

(45.7)
$$\begin{aligned}
&b^{\prec^{s,u}}(v) \leq 0 \text{ if } s \preceq v \prec u,\\
&b^{\prec^{s,u}}(v) \geq 0 \text{ if } v = u,\\
&b^{\prec^{s,u}}(v) = 0 \text{ otherwise.}
\end{aligned}$$

To prove this, observe that if $T \subseteq U \subseteq S$, then for any $v \in S \setminus U$ we have by the submodularity of f:

(45.8)
$$f(U \cup \{v\}) - f(U) \leq f(T \cup \{v\}) - f(T).$$

To see (45.7), if $s \preceq v \prec u$, then by (45.8),

(45.9)
$$\begin{aligned}
&b^{\prec^{s,u}}(v) = f(v_{\prec^{s,u}} \cup \{v\}) - f(v_{\prec^{s,u}}) \leq f(v_{\prec} \cup \{v\}) - f(v_{\prec})\\
&= b^{\prec}(v) = 0,
\end{aligned}$$

since $v_{\prec^{s,u}} = v_{\prec} \cup \{u\} \supset v_{\prec}$.

Similarly,

(45.10)
$$\begin{aligned}
&b^{\prec^{s,u}}(u) = f(u_{\prec^{s,u}} \cup \{u\}) - f(u_{\prec^{s,u}}) \geq f(u_{\prec} \cup \{u\}) - f(u_{\prec})\\
&= b^{\prec}(u) = 0,
\end{aligned}$$

since $u_{\prec^{s,u}} = s_{\prec} \subset u_{\prec}$.

Finally, if $v \prec s$ or $u \prec v$, then $v_{\prec^{s,u}} = v_{\prec}$, and hence $b^{\prec^{s,u}}(v) = b^{\prec}(v) = 0$. This shows (45.7).

By (45.7), the matrix $M = (b^{\prec^{s,u}}(v))_{u,v}$ with rows indexed by $u \in (s,t]_{\prec}$ and columns indexed by $v \in S$, in the order given by $\prec$, has the following, partially triangular, shape, where $+$ means that the entry is ≥ 0, and $-$ that the entry is ≤ 0:

$$\begin{array}{c|cccccccccccccc}
 & & & & s & & & & & & & t & & & \\
\hline
 & 0 & \cdots & 0 & - & + & 0 & \cdots & \cdots & \cdots & 0 & 0 & 0 & \cdots & 0\\
 & \vdots & & \vdots & - & - & + & \ddots & & & \vdots & \vdots & \vdots & & \vdots\\
 & \vdots & & \vdots & - & - & - & \ddots & \ddots & & \vdots & \vdots & \vdots & & \vdots\\
 & \vdots & & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots & & \vdots\\
 & \vdots & & \vdots & \vdots & \vdots & \vdots & & \ddots & \ddots & 0 & 0 & \vdots & & \vdots\\
 & \vdots & & \vdots & \vdots & \vdots & \vdots & & & \ddots & + & 0 & \vdots & & \vdots\\
t & 0 & \cdots & 0 & - & - & - & \cdots & \cdots & \cdots & - & + & 0 & \cdots & 0
\end{array}$$

As each row of M represents a vector $b^{\prec^{s,u}}$, to obtain (45.6) we must describe $\delta(\chi^t - \chi^s)$ as a convex combination of the rows of M, for some $\delta \geq 0$.

We call the $+$ entries in the matrix the 'diagonal' elements. Now for each row of M, the sum of its entries is 0, as $b^{\prec^{s,u}}(S) = f(S) = b^{\prec}(S) = 0$. Hence, if a 'diagonal' element $b^{\prec^{s,u}}(u)$ is equal to 0 for some $u \in (s,t]_{\prec}$, then the corresponding row of M is all-zero. So in this case we can take $\delta = 0$ in (45.6).

If $b^{\prec^{s,u}}(u) > 0$ for each $u \in (s,t]_{\prec}$ (that is, if each 'diagonal' element is strictly positive), then $\chi^t - \chi^s$ can be described as a nonnegative combination of the rows of M (by the sign pattern of M and since the entries in each row of M add up to 0). Hence $\delta(\chi^t - \chi^s)$ is a convex combination of the rows of M for some $\delta > 0$, yielding again (45.6).

45.4. Minimizing a submodular function

We now describe the algorithm to find the minimum value of a submodular set function f on S. We assume $f(\emptyset) = 0$ and $S = \{1, \ldots, n\}$.

We iteratively update a vector $x \in B_f$, given as a convex combination

(45.11) $$x = \lambda_1 b^{\prec_1} + \cdots + \lambda_k b^{\prec_k},$$

where the $\prec_i$ are total orders of S, and where the λ_i are positive and sum to 1. Initially, we choose an arbitrary total order $\prec$ and set $x := b^{\prec}$ (so $k = 1$ and $\prec_1 = \prec$).

We describe the iteration. Consider the directed graph $D = (S, A)$, with

(45.12) $$A := \{(u,v) \mid \exists i = 1, \ldots, k : u \prec_i v\}.$$

Define

(45.13) $$P := \{v \in S \mid x(v) > 0\} \text{ and } N := \{v \in S \mid x(v) < 0\}.$$

Case 1: *D* has no path from *P* to *N*. Then let U be the set of vertices of D that can reach N by a directed path. So $N \subseteq U$ and $U \cap P = \emptyset$; that is, U contains all negative components of x and no positive components. Hence $x(W) \geq x(U)$ for each $W \subseteq S$. As no arcs of D enter U, U is a lower ideal of $\prec_i$, and hence $b^{\prec_i}(U) = f(U)$, for each $i = 1, \ldots, k$. Therefore, for each $W \subseteq S$:

(45.14) $$f(U) = \sum_{i=1}^{k} \lambda_i b^{\prec_i}(U) = x(U) \leq x(W) \leq f(W).$$

So U minimizes f.

Case 2: *D* has a path from *P* to *N*. Let $d(v)$ denote the distance in D from P to v (= minimum number of arcs in a directed path from P to v). Set $d(v) := \infty$ if v is not reachable from P. Choose $s, t \in S$ as follows.

Let t be the element in N reachable from P with $d(t)$ maximum, such that t is largest. Let s be the element with $(s,t) \in A$, $d(s) = d(t) - 1$, and s largest. Let α be the maximum of $|(s,t]_{\prec_i}|$ over $i = 1, \ldots, k$. Reorder indices such that $|(s,t]_{\prec_1}| = \alpha$.

By (45.6), we can find $\delta \geq 0$ and describe

(45.15) $\qquad b^{\prec_1} + \delta(\chi^t - \chi^s)$

as a convex combination of the $b^{\prec_1^{s,u}}$ for $u \in (s,t]_{\prec_1}$. Then with (45.11) we obtain

(45.16) $\qquad y := x + \lambda_1 \delta(\chi^t - \chi^s)$

as a convex combination of $b^{\prec_i}$ $(i = 2, \ldots, k)$ and $b^{\prec_1^{s,u}}$ $(u \in (s,t]_{\prec_1})$.

Let x' be the point on the line segment $\overline{xy}$ closest to y satisfying $x'(t) \leq 0$. (So $x'(t) = 0$ or $x' = y$.) We can describe x' as a convex combination of $b^{\prec_i}$ $(i = 1, \ldots, k)$ and $b^{\prec_1^{s,u}}$ $(u \in (s,t]_{\prec_1})$. Moreover, if $x'(t) < 0$, then we can do without $b^{\prec_1}$.

We reduce the number of terms in the convex decomposition of x' to at most $|S|$ by linear algebra: any affine dependence of the vectors in the decomposition yields a reduction of the number of terms in the decomposition, as in the standard proof of Carathéodory's theorem (subtract an appropriate multiple of the linear expression giving the affine dependence, from the linear expression giving the convex combination, such that all coefficients remain nonnegative, and at least one becomes 0). As all $b^{\prec}$ belong to a hyperplane, this reduces the number of terms to at most $|S|$.

Then reset $x := x'$ and iterate. This finishes the description of the algorithm.

45.5. Running time of the algorithm

We show that the number of iterations is at most $|S|^6$. Consider any iteration. Let

(45.17) $\qquad \beta :=$ number of $i \in \{1, \ldots, k\}$ with $|(s,t]_{\prec_i}| = \alpha$.

Let x', d', A', P', N', t', s', α', β' be the objects x, d, A, P, N, t, s, α, β in the next iteration (if any). Then

(45.18) $\qquad$ for all $v \in S$, $d'(v) \geq d(v)$,

and

(45.19) $\qquad$ if $d'(v) = d(v)$ for all $v \in S$, then $(d'(t'), t', s', \alpha', \beta')$ is lexicographically less than $(d(t), t, s, \alpha, \beta)$.

Since each of $d(t), t, s, \alpha, \beta$ is at most $|S|$, and since (if $d(v)$ is unchanged for all v) there are at most $|S|$ pairs $(d(t), t)$, (45.19) implies that in at most $|S|^4$

iterations $d(v)$ increases for at least one v. Any fixed v can have at most $|S|$ such increases, and hence the number of iterations is at most $|S|^6$.

In order to prove (45.18) and (45.19), notice that

(45.20) for each arc $(v, w) \in A' \setminus A$ we have $s \preceq_1 w \prec_1 v \preceq_1 t$.

Indeed, as $(v, w) \notin A$ we have $w \prec_1 v$. As $(v, w) \in A'$, we have $v \prec_1^{s,u} w$ for some $u \in (s, t]_{\prec_1}$. Hence the definition of $\prec_1^{s,u}$ gives $v = u$ and $s \preceq_1 w \prec_1 u$. This shows (45.20).

If (45.18) does not hold, then $A' \setminus A$ contains an arc (v, w) with $d(w) \geq d(v) + 2$ (using that $P' \subseteq P$). By (45.20), $s \preceq_1 w \prec_1 v \preceq_1 t$, and so $d(w) \leq d(s) + 1 = d(t) \leq d(v) + 1$, a contradiction. This shows (45.18).

To prove (45.19), assume that $d'(v) = d(v)$ for all $v \in S$. As $x'(t') < 0$, we have $x(t') < 0$ or $t' = s$. So by our criterion for choosing t (maximizing $(d(t), t)$ lexicographically), and since $d(s) < d(t)$, we know that $d(t') \leq d(t)$, and that if $d(t') = d(t)$, then $t' \leq t$.

Next assume that moreover $d(t') = d(t)$ and $t' = t$. As $(s', t) \in A'$, and as (by (45.20)) $A' \setminus A$ contains no arc entering t, we have $(s', t) \in A$, and so $s' \leq s$, by the maximality of s.

Finally assume that moreover $s' = s$. As $(s, t]_{\prec_1^{s,u}}$ is a proper subset of $(s, t]_{\prec_1}$ for each $u \in (s, t]_{\prec_1}$, we know that $\alpha' \leq \alpha$. Moreover, if $\alpha' = \alpha$, then $\beta' < \beta$, since $\prec_1$ does not occur anymore among the linear orders making the convex combination, as $x'(t) < 0$. This proves (45.19).

We therefore have proved:

Theorem 45.1. *Given a submodular function f by a value giving oracle, a set U minimizing $f(U)$ can be found in strongly polynomial time.*

Proof. See above. ∎

This algorithm performs the elementary arithmetic operations on function values, including multiplication and division (in order to solve certain systems of linear equations). One would wish to have a 'fully combinatorial' algorithm, in which the function values are only compared, added, and subtracted. The existence of such an algorithm was shown by Iwata [2002a, 2002c], by extending the algorithm of Iwata, Fleischer, and Fujishige [2000, 2001].

Notes. In the algorithm, we have chosen t and s largest possible, in some fixed order of S. To obtain the above running time bound it only suffices to choose t and s in a consistent way. That is, if the set of choices for t is the same as in the previous iteration, then we should choose the same t — and similarly for s. This roots in the idea of 'consistent breadth-first search' of Schönsleben [1980].

The observation that the number of iterations in the algorithm of Section 45.4 is $O(|S|^6)$ instead of $O(|S|^7)$ is due to L.K. Fleischer. Vygen [2002] showed that the number of iterations can in fact be bounded by $O(|S|^5)$.

The algorithm described above has been speeded up by Fleischer and Iwata [2000,2002], by incorporating a push-relabel type of iteration based on approximate distances instead of precise distances (like Goldberg's method for maximum flow, given in Section 10.7). Iwata [2002b] combined the approaches of Iwata, Fleischer, and Fujishige [2000,2001] and Schrijver [2000a] to obtain a faster algorithm. A descent method for submodular function minimization based on an oracle for membership of the base polytope was given by Fujishige and Iwata [2002].

Surveys and background on submodular function minimization are given by Fleischer [2000b] and McCormick [2001].

45.6. Minimizing a symmetric submodular function

A set function f on S is called *symmetric* if $f(U) = f(S \setminus U)$ for each $U \subseteq S$. The minimum of a symmetric submodular function f is attained by $\emptyset$, since for each $U \subseteq S$ one has

(45.21) $$2f(U) = f(U) + f(S \setminus U) \geq f(\emptyset) + f(S) = 2f(\emptyset).$$

By extending a method of Nagamochi and Ibaraki [1992b] for finding the minimum nonempty cut in an undirected graph, Queyranne [1995,1998] gave an easy combinatorial algorithm to find a nonempty proper subset U of S minimizing $f(U)$, where f is given by a value giving oracle. We may assume that $f(\emptyset) = f(S) = 0$, by resetting $f(U) := f(U) - f(\emptyset)$ for all $U \subseteq S$.

Call an ordering $s_1, \ldots, s_n$ of the elements of S a *legal order* of S for f, if, for each $i = 1, \ldots, n$,

(45.22) $$f(\{s_1, \ldots, s_{i-1}, x\}) - f(\{x\})$$

is minimized over $x \in S \setminus \{s_1, \ldots, s_{i-1}\}$ by $x = s_i$. One easily finds a legal order, by $O(|S|^2)$ oracle calls (for the value of f).

Now the algorithm is (where a set U *splits* a set X if both $X \cap U$ and $X \setminus U$ are nonempty):

(45.23) Find a legal order $(s_1, \ldots, s_n)$ of S for f.
Determine (recursively) a nonempty proper subset T of S not splitting $\{s_{n-1}, s_n\}$, minimizing $f(T)$. (This can be done by identifying s_{n-1} and s_n.)
Then the minimum value of $f(U)$ over nonempty proper subsets U of S is equal to $\min\{f(T), f(\{s_n\})\}$.

The correctness of the algorithm follows from, for $n \geq 2$:

(45.24) $f(U) \geq f(\{s_n\})$ for each $U \subseteq S$ splitting $\{s_{n-1}, s_n\}$.

This can be proved as follows. Define $t_0 := s_1$. For $i = 1, \ldots, n-1$, define $t_i := s_j$, where j is the smallest index such that $j > i$ and such that U splits $\{s_i, s_j\}$. For $i = 0, \ldots, n$, let $U_i := \{s_1, \ldots, s_i\}$. Note that for each $i = 1, \ldots, n-1$ one has

(45.25) $$f(U_{i-1} \cup \{t_i\}) - f(\{t_i\}) \geq f(U_{i-1} \cup \{t_{i-1}\}) - f(\{t_{i-1}\}),$$

since if $t_{i-1} = t_i$ this is trivial, and if $t_{i-1} \neq t_i$, then $t_{i-1} = s_i$, in which case (45.25) follows from the legality of the order.

Moreover, for each $i = 1, \ldots, n-1$ (setting $\overline{U} := S \setminus U$):

(45.26) $$\begin{aligned} &f(U_i \cup U) - f(U_{i-1} \cup U) + f(U_i \cup \overline{U}) - f(U_{i-1} \cup \overline{U}) \\ &\leq f(U_i \cup \{t_i\}) - f(U_{i-1} \cup \{t_i\}). \end{aligned}$$

In proving this, we may assume (by symmetry of U and $\overline{U}$) that $s_i \in \overline{U}$. Then $U_i \cup \overline{U} = U_{i-1} \cup \overline{U}$ and $t_i \in U$. So $f(U_i \cup \{t_i\}) + f(U_{i-1} \cup U) \geq f(U_{i-1} \cup \{t_i\}) + f(U_i \cup U)$, by submodularity. This gives (45.26).

Then we have:

(45.27) $$\begin{aligned} &f(s_n) - 2f(U) \\ &= f(U_{n-1} \cup U) + f(U_{n-1} \cup \overline{U}) - f(U_0 \cup U) - f(U_0 \cup \overline{U}) \\ &= \sum_{i=1}^{n-1} (f(U_i \cup U) - f(U_{i-1} \cup U) + f(U_i \cup \overline{U}) - f(U_{i-1} \cup \overline{U})) \\ &\leq \sum_{i=1}^{n-1} (f(U_i \cup \{t_i\}) - f(U_{i-1} \cup \{t_i\})) \\ &\leq \sum_{i=1}^{n-1} (f(U_i \cup \{t_i\}) - f(U_{i-1} \cup \{t_{i-1}\}) + f(\{t_{i-1}\}) - f(\{t_i\})) \\ &= f(U_{n-1} \cup \{t_{n-1}\}) - f(\{t_{n-1}\}) - f(\{t_0\}) + f(\{t_0\}) = -f(s_n) \end{aligned}$$

(where the first inequality follows from (45.26), and the second inequality from (45.25)). This shows (45.24).

Notes. Fujishige [1998] gave an alternative correctness proof. Nagamochi and Ibaraki [1998] extended the algorithm to minimizing submodular functions f satisfying

(45.28) $$f(T) + f(U) \geq f(T \setminus U) + f(U \setminus T)$$

for all $T, U \subseteq S$. Rizzi [2000b] gave an extension.

45.7. Minimizing a submodular function over the odd sets

From the strong polynomial-time solvability of submodular function minimization, one can derive that also a set of odd cardinality minimizing f (over the odd sets) is solvable in strongly polynomial time (Grötschel, Lovász, and Schrijver [1981,1984a,1988] (the second paper corrects a wrong argument given in the first paper)).

Theorem 45.2. *Given a submodular set function f on S (by a value giving oracle) and a nonempty subset T of S, one can find in strongly polynomial time a set $W \subseteq S$ minimizing $f(W)$ over W with $|W \cap T|$ odd.*

Proof. The case T odd can be reduced to the case T even as follows. Find for each $t \in T$ a subset W_t of $S - t$ with $W_t \cap (T - t)$ odd, and minimizing $f(W_t)$. Moreover, find a subset U of S minimizing $f(U)$ over $U \supseteq T$. Then a set that attains the minimum among $f(U)$ and the $f(W_t)$, is an output as required.

So we can assume that T is even. We describe a recursive algorithm. Say that a set U *splits* T if both $T \cap U$ and $T \setminus U$ are nonempty. First find a set U minimizing $f(U)$ over all subsets U of S splitting T. This can be done by finding for all $s, t \in T$ a set $U_{s,t}$ minimizing $f(U_{s,t})$ over all subsets of S containing s and not containing t (this amounts to submodular function minimization), and taking for U a set that minimizes $f(U_{s,t})$ over all such s, t.

If $U \cap T$ is odd, we output $W := U$. If $U \cap T$ is even, then recursively we find a set X minimizing $f(X)$ over all X with $X \cap (T \cap U)$ odd, and not splitting $T \setminus U$. This can be done by shrinking $T \setminus U$ to one element. Also, recursively we can find a set Y minimizing $f(Y)$ over all Y with $Y \cap (T \setminus U)$ odd, and not splitting $T \cap U$. Output an X or Y attaining the minimum of $f(X)$ and $f(Y)$.

This gives a strongly polynomial-time algorithm as the total number of recursive calls is at most $|T| - 2$ (since $2 + (|T \cap U| - 2) + (|T \setminus U| - 2) = |T| - 2$).

To see the correctness, let W minimize $f(W)$ over those W with $|W \cap T|$ odd. Suppose that $f(W) < f(X)$ and $f(W) < f(Y)$. As $f(W) < f(X)$, W splits $T \setminus U$, and hence $W \cup U$ splits T. Similarly, $f(W) < f(Y)$ implies that $W \cap U$ splits T.

Since $W \cap T$ is odd and $U \cap T$ is even, either $(W \cap U) \cap T$ or $(W \cup U) \cap T$ is odd.

If $(W \cap U) \cap T$ is odd, then $f(W \cap U) \geq f(W)$ (as W minimizes $f(W)$ over W with $W \cap T$ odd) and $f(W \cup U) \geq f(U)$ (as $W \cup U$ splits T and as U minimizes $f(U)$ over U splitting T). Hence, by the submodularity of f, $f(W \cap U) = f(W)$. Since $(W \cap U) \cap (T \cap U) = (W \cap U) \cap T$ is odd and since $W \cap U$ does not split $T \setminus U$, we have $f(W) = f(W \cap U) \geq f(X)$, contradicting our assumption.

If $(W \cup U) \cap T$ is odd, a symmetric argument gives a contradiction. ▌

This generalizes the strong polynomial-time solvability of finding a minimum-capacity odd cut in a graph, proved by Padberg and Rao [1982] (Corollary 25.6a). For a further generalization, see Section 49.11a.