Lecture 16

Submodular Functions and Network Coding

Nov 22, 2004
Lecturer: Kamal Jain
Notes: Atri Rudra

In the last lecture we looked at submodular function minimization. We will present a recent result on submodular functions in this lecture and then move onto the question if the max flow min cut theorem still holds when the number of receivers are not all the nodes (which was same as arborescence packing) or just one node (which was same as max flow).

16.1 Submodular function minimization

In this section, we look at the question of giving an NP certificate for a subset S which supposedly minimizes the submodular function f whose domain is 2^U for some universe U. Note that giving a coNP certificate is easy as one can obtain an optimal solution (using the z_is as discussed in the last lecture).

The following result is from [1].

Lemma 16.1. *For any permutation π of U, we have*

$$x_{\pi_1} f(\{\pi_1, \cdots, \pi_n\}) + (x_{\pi_2} - x_{\pi_1}) f(\{\pi_2, \cdots \pi_n\}) + \cdots + (x_{\pi_n} - x_{\pi_{n-1}}) f(\{\pi_n\}) + (1 - x_{\pi_n}) f(\{\}) \leq f(x_1, \cdots, x_n)$$

The inequality is tight when π orders the x_is in ascending order. For notational convenience we will refer to the LHS of the inequality as $E_f(\pi, \boldsymbol{x})$.

Proof. First notice that when π orders x_is in increasing order then x_{π_i} is the same as the z_is defined for an optimal solution for the problem in the last lecture. Thus, in this case the inequality is tight.

To complete the proof we consider a π that does not order the x_is in ascending order. Thus, there exists a j such that $x_{\pi_j} > x_{\pi_{j+1}}$. Define another permutation π' such that $\pi'_j = \pi_{j+1}$, $\pi'_{j+1} = \pi_j$ and $\pi'_i = \pi_i$ for $i \notin \{j, j+1\}$. We will show that $E_f(\pi', \boldsymbol{x}) > E_f(\pi, \boldsymbol{x})$ which with the discussion in the last paragraph would complete the proof.

We will work with the quantity $E_f(\pi', \boldsymbol{x}) - E_f(\pi, \boldsymbol{x})$. We have $E_f(\pi', \boldsymbol{x}) - E_f(\pi, \boldsymbol{x}) =$
$x_{\pi'_j} f(\{\pi'_j, \pi'_{j+1}, \pi_{j+2}, \cdots, \pi_n\}) - x_{\pi'_j} f(\{\pi'_{j+1}, \pi_{j+2}, \cdots, \pi_n\}) + x_{\pi'_{j+1}} f(\{\pi'_{j+1}, \pi_{j+2}, \cdots, \pi_n\}) - x_{\pi'_{j+1}} f(\{\pi_{j+2}, \cdots, \pi_n\}) - [x_{\pi_j} f(\{\pi_j, \pi_{j+1}, \pi_{j+2}, \cdots, \pi_n\}) - x_{\pi_j} f(\{\pi_{j+1}, \pi_{j+2}, \cdots, \pi_n\}) + x_{\pi_{j+1}} f(\{\pi_{j+1}, \pi_{j+2}, \cdots, \pi_n\}) - x_{\pi_{j+1}} f(\{\pi_{j+2}, \cdots, \pi_n\})]$
$= x_{\pi_{j+1}} f(\{\pi_{j+1}, \pi_j, \pi_{j+2}, \cdots, \pi_n\}) - x_{\pi_{j+1}} f(\{\pi_j, \pi_{j+2}, \cdots, \pi_n\}) + x_{\pi_j} f(\{\pi_j, \pi_{j+2}, \cdots, \pi_n\}) -$

$x_{\pi_j} f(\{\pi_{j+2}, \cdots, \pi_n\}) - [x_{\pi_j} f(\{\pi_j, \pi_{j+1}, \pi_{j+2}, \cdots, \pi_n\}) - x_{\pi_j} f(\{\pi_{j+1}, \pi_{j+2}, \cdots, \pi_n\}) + x_{\pi_{j+1}} f(\{\pi_{j+1}, \pi_{j+2}, \cdots, \pi_n\}) - x_{\pi_{j+1}} f(\{\pi_{j+2}, \cdots, \pi_n\})]$
$= -(x_{\pi_j} - x_{\pi_{j+1}}) f(\{\pi_j, \pi_{j+1}, \pi_{j+2}, \cdots, \pi_n\}) + (x_{\pi_j} - x_{\pi_{j+1}}) f(\{\pi_j, \pi_{j+2}, \cdots, \pi_n\}) + (x_{\pi_j} - x_{\pi_{j+1}}) f(\{\pi_{j+1}, \pi_{j+2}, \cdots, \pi_n\}) - (x_{\pi_j} - x_{\pi_{j+1}}) f(\{\pi_{j+2}, \cdots, \pi_n\})$
$= (x_{\pi_j} - x_{\pi_{j+1}})[f(\{\pi_j, \pi_{j+2}, \cdots, \pi_n\}) + f(\{\pi_{j+1}, \pi_{j+2}, \cdots, \pi_n\}) - f(\{\pi_j, \pi_{j+1}, \pi_{j+2}, \cdots, \pi_n\}) - f(\{\pi_{j+2}, \cdots, \pi_n\})]$.
The quantity in the first pair of parenthesis is positive by our assumption on π while the second term is non-negative by the submodualrity of f. Thus, we have $E_f(\pi', \boldsymbol{x}) > E_f(\pi, \boldsymbol{x})$ and we are done. □

We can now express our LP for minimizing a submodular function f by

$$min \ b$$

subject to

$$\forall \pi \quad E_f(\pi, \boldsymbol{x}) \leq b$$
$$0 \leq x_i \leq 1$$

If we now consider the dual of the above LP, then it has exponentially many variables and polynomially many constraints which implies polynomially many variables would be non-zero in the optimal dual solution. These non-zero variables constitute the NP certificate we were looking for.

16.2 Errata for last lecture

We give a much simpler argument than the one used in the last lecture to show that one can build abrorescences inductively. As a quick recap let G be the original graph and let A be the partially built arborescence where we always maintain the relation $C(G - A) \geq C(G) - 1$ where $C(G)$ is the min cut of G. Further we defined a critical set $S \subseteq V(G)$ as one which satisfies the following three properties:

1. $\delta^{G-A}(S) = C(G) - 1$,
2. The root is in S, and
3. There exists a $w \in V(G) - V(A)$ such that $w \notin S$.

Let S' be a maximal critical set. Define $T = S \cup V(A)$. Thus, by maximality of S' we have that $\delta^{G-A}(T) = C(G)$ and hence, there exists an edge $e = (u, v)$ such that both u and v lie outside of $V(A)$. Now, if we could pick this e such that adding e to A does not affect any critical set then we are done. So assume we have a critical set T' such that $u \in T'$ and $v \notin T'$: that is, removing e would affect the critical set T'. Now, by submodualrity of the $\delta^{G-A}(.)$ function, we have

$$\delta^{G-A}(T') + \delta^{G-A}(S') \geq \delta^{G-A}(S' \cap T') + \delta^{G-A}(S' \cup T'). \tag{16.1}$$

As both S' and T' are critical sets, $\delta^{G-A}(T') = \delta^{G-A}(S') = C(G) - 1$. Further, as we always maintain $C(G - A) \geq C(G) - 1$, we have $\delta^{G-A}(S' \cap T') \geq C(G) - 1$ and $\delta^{G-A}(S' \cup T') \geq C(G) - 1$. These along with (16.1) implies that $\delta^{G-A}(S' \cup T') = C(G) - 1$, that is, $S' \cup T'$ is a critical set which contradicts the maximality of S'.

16.3 Network Coding

We have seen that when there is a single source and a single sink then max flow is the same as min cut. On the other extreme when we have a single source and all other nodes in the graphs are sinks then again the maximum flow (according to the arborescence packing) is same as the min cut. It is a natural question to ask if this generalizes to the intermediate case: that is, when not every node other than the source is a sink. Note that in this case we are taking about packing of Steiner trees and it is easy to convince yourself that max flow in this case is upper bounded by the min cut. However, the other direction is not true. Consider the example in Figure 16.1.

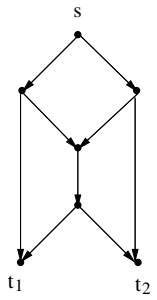

Figure 16.1: Example in Steiner packing where max flow is not the same as min cut. s is the source and t_1 and t_2 are the sinks.

It is easy to see that the graph in Figure 16.1 has minimum cut of two (two edge disjoint paths to t_1 and t_2). However, one cannot pack more than one integral Steiner tree. If one is allowed to do fraction packing then one can pack atmost 1.5 steiner trees. This latter fact can be proved using the dual but we'll just give an fractional packing of 1.5 here. Consider the Steiner trees in Figure 16.2. If we take each tree to the extent

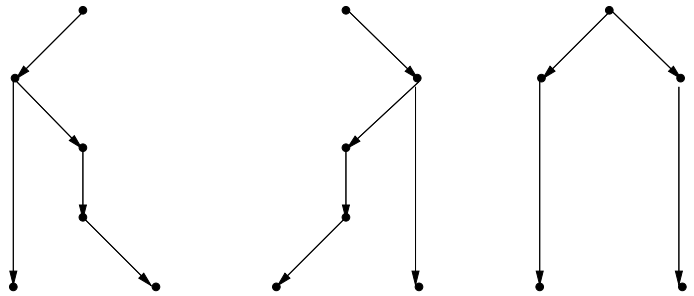

Figure 16.2: Steiner Trees used in the fractional packing of the graph in figure 16.1.

of $\frac{1}{2}$, then we get a packing of 1.5.

Again consider the example in of Figure 16.1 as shown in Figure 16.3 (the example is due to [2]). At at each time instance, two bits b_1 and b_2 originate at the source. If we do not do any computation on the edges then c would be either b_1 (or b_2) in which case the only the second (first) sink gets both the bits. However, if $c = b_1 \oplus b_2$, then both the sinks can recover b_1 and b_2.

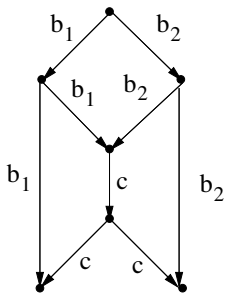

Figure 16.3: Same example as Figure 16.1 but with computation being done at the edges. For example consider $c = b_1 \oplus b_2$.

With the above motivating example, we will try to give a general framework where we can do computation on the edges. However, we will have to work over general field $GF(q)$ instead of $GF(2)$. So suppose we have a digraph G one node of which is designated as source which has to transmit c $GF(q)$ elements at each time step to some receivers (or sinks). We also assume that there are c edge disjoint paths from the source to each receiver (paths for different receivers may not be disjoint). Further the edges have a capacity of $\log q$ bits, that is, they can transmit a $GF(q)$ element at each time instance. One final assumption on the graph is that it is acyclic. This is not a very strong assumption: any graph with cycles can be converted into a larger acyclic graph by "unrolling" [2]. For the rest of this course we will assume that G is acyclic.

The original solution in [2], picks a random function from $GF(q)$ to $GF(q)$ to do computation on the edges and they show that with exponentially small probability, each receiver can 'decode' the original c information symbols transmitted from the source. Thus, by a simple union bound, the probability that some receiver fails to decode is still exponentially small.

We now outline a solution which specifies how to compute on each edge of the graph G so that if the source s sends symbols $b_1, \cdots, b_c \in GF(q)$ every time unit then all the receivers $r_1, \cdots, r_t$ can decode $b_1, \cdots, b_c$ from the symbols that they receive. The solution is basically solving an "algebraic program". For any two edges e and f in G such that e is an incoming edge and f is an outgoing edge for some node in G, define a variable α_{ef} and let $\overline{\alpha}$ be the vector of all such variables. Also let b_e be the symbol being transmitted on edge $e \in E(G)$. We will do a linear encoding at each edge. Let $v \in V(G)$. Further, let $e_1, \cdots, e_m \in E(G)$ be the set of incoming edges of v and let $f \in E(G)$ be an outgoing edge of v. Then $b_e = \sum_{i=1}^{m} \alpha_{e_i f} b_{e_i}$. Now consider any receiver r_i and wlog let $e_1^i, \cdots, e_c^i$ be the set of incoming edges of r_i. Given the linear combination that we do at each edge, it is easy to see that for $1 \leq j \leq c$, $b_{e_j^i} = \sum_{k=1}^{c} P_k^j(\overline{\alpha}) b_k$ where P_k^j is some polynomial over $GF(q)$ in the variables of $\overline{\alpha}$. We now define a $c \times c$ matrix $M^{(i)}$ such that $M_{jk}^{(i)} = P_k^j(\overline{\alpha})$. Thus, receiver r_i can decode and obtain $b_1, \cdots, b_c$ if $det(M^{(i)}) \neq 0$. Note that $det(M^{(i)})$ is equivalent to some polynomial $P^{(i)}(\overline{\alpha})$. Thus, we finally want to solve the following algebraic program

$$\forall i \leq i \leq t, P^{(i)}(\overline{\alpha}) \neq 0$$

If we choose q to be greater than the sum of degree of all $P^{(i)}$ for $1 \leq i \leq t$, then the above algebraic program is feasible if and only if

$$\prod_{i=1}^{t} P^{(i)}(\overline{\alpha}) \neq 0$$

We now argue that it is always possible to find α such that $P^{(i)}(\overline{\alpha}) \neq 0$ for some particular i: indeed α as a feasible c-flow between the source and the receiver r_i is one such solution. Finally, we use the second formulation to derive one value of $\overline{\alpha}$ that works for all receivers– good probabilistic algorithms for finding non-roots of a polynomial over $GF(q)$ are well known[1].

In the scheme presented above, the size of q depends on the capacity of the network which can be pretty bad. In the next lecture we will present a deterministic scheme where $q = O(t)$.

References

[1] K. Jain, V. Vazirani, and Y. Ye. Market equilibria for homothetic, quasi-concave utilities and economics of scale in production. In *SODA 05*, 2005. To Appear.

[2] R. Ahlswede, N. Kai, S. Y. R. Li, and R. Yeung Network Information Flow. In *IEEE Trans. Inform. Theory*, IT-46: 1204-1216, 2000.

[1] In fact a random point in $GF(q)$ is not a root of a polynomial with probability which depends only on the degree of the polynomial and the field size: the well known Schwartz Zippel lemma.