CMPUT 675: Topics in Algorithms and Combinatorial Optimization (Fall 2009)

Lecture 10,11: General Matching Polytope, Maximum Flow

Lecturer: Mohammad R. Salavatipour **Scriber:** Mohammad Reza Khani
Date: Oct 6 and 8, 2009

In this lecture we finish the discussion of matching (and perfect matching) polytopes. We prove that the matching polytope is integral. Then we start the topic of flows and circulations.

Recall that if $A = |V| \times |E|$ is the incident vector of a graph G, then A is totally unimodular (TUM) if and only if G is bipartite. We also proved that for every TUM matrix A, and integer vector b, the polytope $Ax \leq b$ is integral. This includes the matching polytope of bipartite graphs.

1 Perfect Matching and Matching Polytope on General Graphs

Let G be a general graph and X be the set of all perfect matchings of G, i.e., all incident vectors in $\mathbb{R}^{|E|}$ where each corresponds to a perfect matching. The perfect matching polytope of G, $P(G)$, is defined as the convex hull of X. Consider the following LP:

$$\begin{aligned} x(\delta(v)) &= 1 \quad \forall v \\ x_e &\geq 0 \end{aligned} \tag{1}$$

It is easy to see that every perfect matching satisfies these constraints. However, these constraints are not sufficient to define correctly a perfect matching. For example, if we consider any odd cycle, say C_3 and assign $\frac{1}{2}$ to each edge, it satisfies these constraints but it does not belong to the perfect matching polytope (as it is empty).

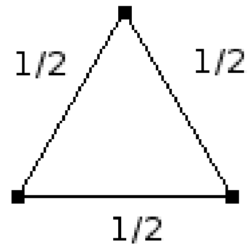

Figure 1: Triangle Counter Example

Let $U \subseteq V$ be an odd size set and M be any perfect Matching of G. **Observation:** M can have at most $\frac{|U|-1}{2}$ edges inside U. Thus, at least one vertex of U is matched outside. This can be used to strengthen the LP by adding "odd set" constraints:

$$
\begin{aligned}
x(\delta(v)) &= 1 \quad \forall v \\
x(E(U)) &\leq \frac{|U|-1}{2} \quad \forall \text{ odd set } U \subseteq V \\
x_e &\geq 0
\end{aligned} \tag{2}
$$

Note that these constraints are violated by the example of Figure 1. Edmonds proved that the LP(2) corresponds to the polytope $P(G)$, for perfect matchings in G.

Theorem 1.1 (Edmonds) *LP(2) is integral, i.e. it is the polytope for perfect matchings of G.*

Proof: Here we prove that all bfs's of this LP are integer, therefore the polytope is integral. We prove this by showing (inductively) that every fractional solution is a non-vertex, i.e. can be written as convex combination of perfect matchings of G. Let x be any basic feasible solution for this LP and suppose that it does not correspond to a perfect matching. First, we can assume that x is totally fractional, i.e. $0 < x_e < 1$ for every edge since if there is an edge e with $x_e = 1$ we remove both the edge and the end points to reduce the problem to a smaller graph, and if $x_e = 0$ we can simply delete that edge and the corresponding variable. So we arrive at reduced graph and reduced bfs solution x which is totally fractional. Note that all values of x are rational. For simplicity, assume that every fractional value is a multiple of $1/k$ (for sufficiently large k this is true). Thus, we can assume that instead of each edge, we have parallel edges each with fractional value $1/k$ and the number of parallel edges we have will be kx_e. So x corresponds to a set of tight constraints. Tight odd-set constraints of single nodes are implied by degree constraints. Therefore, we can assume every odd-set constraint that is tight is for a set U with $3 \leq |U| \leq |V| - 3$, and for that we have $x(\delta(U)) = 1$. Consider any such tight set U. We break the graph into two smaller ones G_1 and G_2 as shown in Figure 2 by: once contracting U to a single node (allowing parallel edges) and once contracting $G - U$ to a single node. Both G_1 and G_2 will have smaller number of nodes because of the bound on the size of U.

Figure 2: Removing an Odd-set Tight Constraint

Now consider projection of x to these two graphs, call it x' and x''. Then x' and x'' satisfy the constraints of the LP for G_1 and G_2 respectively, and hence belong to the perfect matching polytopes of G_1 and G_2. So G_1 has perfect matchings $M'_1, M'_2, \ldots, M'_k$ and G_2 has perfect matchings $M''_1, M''_2, \ldots, M''_k$ with

$$x' = \frac{1}{k} \sum_{i=1}^{k} \chi^{M'_i} \quad \text{and} \quad x'' = \frac{1}{k} \sum_{i=1}^{k} \chi^{M''_i}.$$

Now for each edge $e \in \delta(U)$, the number of i with $e \in M'_i$ is equal to $kx'(e) = kx(e) = kx''(e)$, which is equal to the number of i with $e \in M''_i$ Hence, we can assume that for each $i = 1, \ldots, k$, M'_i and M''_i have an edge in $\delta(U)$ in common. So $M_i = M'_i \cup M''_i$ is a perfect matching of G. Thus,

$$x = \frac{1}{k}\sum_{i=1}^{k} \chi^{M_i}.$$

Therefore, after removing edges with values 0/1 and contracting of odd-set constraints, we have only tight degree constraints. If we have $2n$ nodes left $2n$ degree constraints have to be tight. Each node has a degree greater than or equal to 2. We have only $2n$ tight constraints (degree constraints) so we can have only $2n$ edges, because this solution is a bfs. After all, we have a graph with $2n$ nodes and $2n$ edges so it is the union of even cycles. We know that in this case (when the graph is bipartite) any fractional solution can be written as convex combination of other perfect matchings, and this completes the proof. ■

Corollary 1.2 (Edmonds Matching Polytope) *For any graph G the matching polytope can be defined by:*

$$\begin{aligned} x(\delta(v)) &= 1 \quad \forall v \\ x(\delta(U)) &\geq 1 \quad \forall \text{ odd set } U \subseteq V \\ x_e &\geq 0 \end{aligned} \tag{3}$$

Proof: Clearly every matching M satisfies 3. If we prove that every vertex of the polytope defined by LP(3) is a matching then we are done. Suppose that x is a bfs of this LP. Build a copy G' from G and for each $v \in V$ make an edge vv'; call this the new graph $\tilde{G} = (\tilde{V}, \tilde{E})$. for each $e \in E$ we have $\tilde{x}_e = \tilde{x}_{e'} = x_e$ and for each edge $vv' \in \tilde{E}$ we define $\tilde{x}(vv') = 1 - x(\delta(v))$. It can be proved (as below) that $\tilde{x}$ is feasible for the perfect matching LP of $\tilde{G}$: degree constraints are satisfied as every node v has $\tilde{x}(\delta(v)) = 1$ now. Also, for each odd set $U \subseteq \tilde{V}$, say $U = W \cup X'$ with $W, X \subseteq V$ we have

$$\tilde{x}(\tilde{\delta}(U)) \geq \tilde{x}(\tilde{\delta}(W \backslash X)) + \tilde{x}(\tilde{\delta}(X' \backslash W'))$$

So we may assume that $W \cap X = \emptyset$ and by symmetry, we may assume that W is odd, thus $X = \emptyset$. So it is enough to show that any odd set $U \subseteq V$ has $\tilde{x}(\tilde{\delta}(U)) \geq 1$. Now

$$\tilde{x}(\tilde{\delta}(U)) + 2\tilde{x}(\tilde{E}(U)) = \sum_{v \in U} \tilde{x}(\tilde{\delta}(v)) = |U|$$

and hence $\tilde{x}(\tilde{\delta}(U)) = |U| - 2\tilde{x}(\tilde{E}(U)) \geq |U| - 2\lfloor \frac{1}{2}|U| \rfloor = 1$. So $\tilde{x}$ belongs to the perfect matching polytope of $\tilde{G}$ and so can be expressed as convex combination of perfect matchings of $\tilde{G}$. Consider the projection of these perfect matchings to G; then x is essentially the convex combination of these matchings (of G). ■

2 Maximum Flows and Minimum Cuts

Suppose we are given a digraph $D = (V, E)$ and two vertices $s, t \in V$. Every edges has an upper bound capacity $u_e : E \rightarrow \mathbb{R} \geq 0$ and a lower bound capacity $l_e : E \rightarrow \mathbb{R} \geq 0$. A function $f : E \rightarrow \mathbb{R} \geq 0$ is called a flow if:

1. $\forall v \in V - \{s, t\}, \quad \sum_{e \in \delta^+(v)} f(e) = \sum_{e \in \delta^-(v)} f(e)$, i.e. we have flow conservation at every node (except the source and sink),

2. $\forall e \in E, \quad l_e \leq f_e \leq u_e$, i.e. capacity constraints are satisfied.

Definition 2.1 *Total value of a flow f, denoted by $|f|$, is the total flow going out of s, which by flow conservation should be the net flow going into t:*

$$\sum_{e\in\delta^+(s)} f(e) - \sum_{e\in\delta^-(s)} f(e) = |f| = \sum_{e\in\delta^-(t)} f(e) - \sum_{e\in\delta^+(t)} f(e)$$

The maximum flow problem is: Given digraph $G(V,E)$ and $l,u : E \to \mathbb{R} \geq 0$, find an $s-t$-flow of largest possible value. Clearly, one can write this problem as an LP:

$$\begin{array}{rrcll} max & f(\delta^+(s)) & - & f(\delta^-(s)) & \\ s.t. & \sum_{e\in\delta^+(v)} f(e) & = & \sum_{e\in\delta^-(v)} f(e) & \forall v \in V - \{s,t\} \\ & l \leq f & \leq & u & \end{array} \quad (4)$$

$$(5)$$

This can be solved using an LP solver. In fact if we consider N to be the vertex-arc incident vector of the graph then N is an $n \times m$ matrix such that for each edge $e = ij$:

$$N(u,e) = \begin{cases} 1 & u = i \\ -1 & u = j \\ 0 & u \notin \{i,j\} \end{cases}$$

N	...	e	...
$\vdots$			
v_1		+1	
$\vdots$			
v_2		-1	
$\vdots$			

then the constraints of the LP can be written as $Ax \leq b$ where

$$A = \begin{pmatrix} N \\ \hline I \\ \hline -I \end{pmatrix}$$

Then the following lemma is fairly easy to prove:

Lemma 2.2 *A is totally unimodular.*

Proof: We prove inductively that the determinant of any submatrix of A is in $\{-1,0,+1\}$. Base case is trivial. Consider any submatrix of A. If there is a row with a single non-zero entry then clearly expanding the determinant around that entry and using induction hypothesis gives the desired result. So we can assume that the submatrix is actually a submatrix of N. Here again if there is a column with a sinle non-zero etnry (which will be a $+1$ or a -1) then we can expand the determinant around that column and use induction. Therefore, the only case left is when every column of the submatrix of N has exactly two non-zero entries, one $+1$ and one -1. The rows of such a submatrix is linearly dependent since the sum of the rows is then zero, thus the determinant is zero. ■

Corollary 2.3 *If b is integer i.e. if u,l are integers then the flow is always integer.*

2.1 Cuts

Definition 2.4 *Given a set $S \subset V$, the cut defined by S is:*

$$\delta^+(S) = \{uv \in E| \quad u \in s, v \in V/S\}$$

The capacity of the cut is $c(\delta^+(S))$, which is the sum of the capacities of the edges going out of S.

Similarly, $\delta^-(S)$ is the set of edges going into S. We say a cut $\delta^+(S)$ is an $s-t$ cut if $s \in S$ and $t \in V - S$. If we have a flow f and S is an $s-t$-cut then $|f| = f(\delta^+(S)) - f(\delta^-(S))$. It is easy to prove that:

Lemma 2.5 *For any flow f and any $s-t$-cut S, $c(S) \geq f(\delta^+(s)) - f(\delta^-(s)$*

Proof:

$$\begin{aligned} |f| = f(\delta^+(S)) - f(\delta^-(S)) &= \sum_{v \in V-t} (f(\delta^+(v)) - f(\delta^-(v))) \\ &= f(\delta^+(S)) - f(\delta^-(S)) \\ &\leq f(\delta^+(S)) \\ &\leq c(S). \end{aligned}$$

■

So we have weak duality:

$$\max_f |f| \leq \min_S c(S)$$

We will show this the strong duality holds as well i.e. $\max_f |f| = \min_S c(S)$.

Flow decomposition: Suppose f is an $s-t$-flow in G. We can write f as linear combination of a polynomial number of $s-t$-path each carrying $\epsilon > 0$ amount of flow. In particular if $|f|$ is integer then each path carries a unit flow. Because, whenever you enter a vertex by a unit flow because of flow conservation) you can exit from that node as well, unless you have arrived at t. Therefore, any walk that starts from s ends at t. Moreover, in the special case when all capacities are 1 all these flow-paths are edge disjoint. So the size of a maximum flow is equal to the number of edge-disjoint paths from s to t.

Theorem 2.6 *The value of a Max-flow is equal to the size of a min-cut in G.*

Proof: Given graph a $G(V, E)$ and flow f on G we construct residual multi-graph $G_f(V, E_f)$ as follows:

- Forward edge: if $(i, j) \in E$ and $f(i, j) < c(i, j)$: $\quad (i, j) \in E_f$ with capacity $c(i, j) - f(i, j)$.
- Backward edge: if $(j, i) \in E$ and $f(j, i) > 0$: $\quad (i, j) \in E_f$ with capacity $f(j, i)$.

Let f be a max-flow in G. Using weak duality, it is sufficient to find a cut S such that $|f| = c(s)$ to prove the strong duality. We first prove that if f is maximum then there is no $s-t$-path in G_f. By way of contradiction, suppose we find an $s-t$ path P in G_f. So it may contains forward edges and backward edges. Let ϵ be:

$$\epsilon = \min_e \begin{cases} c_e - f_e, & \text{if } e \text{ is a forward edge} \\ f_e, & \text{if } e \text{ is a backward edge} \end{cases}$$

Let p^+ be the set of forward edges and p^- be the set of backward edges then we can define f' to be:

$$f' = \begin{cases} f_e + \epsilon & e \in p^+ \\ f_e - \epsilon & e \in p^- \\ f_e & \text{other wise} \end{cases}$$

Clearly f' is a feasible flow and $|f'| = |f| + \epsilon$, so it contradicts our assumption that f is maximum. Thus there is no path from s to t in G_f. Let S be the set of vertices reachable from s in G_f; clearly $t \notin S$. As there is no $s-t$-path we have following:

$$\forall e \in \delta^+(S): \quad f(e) = c(e)$$
$$\forall e \in \delta^-(S): \quad f(e) = 0$$
$$\text{therefore} \qquad c(S) = \textstyle\sum_{e \in \delta^+(S)} c(e) = \sum_{e \in \delta^+(S)} f(e) = |f|$$

■

This proof suggests the following algorithm to find a maximum flow:
as long as there is an augmenting path in G_f find one and improve the flow and update G_f.

Theorem 2.7 *If all the capacities are rational the above algorithm terminates.*

Proof: Multiply all capacities by sufficiently large integer k so that all c_e's will be integer; the theorem follows since every path increases the flow by at least 1. ■

For irrational values, there are examples which show that the algorithm may never terminate if the augmenting paths are not selected carefully.

Another algorithm introduced by Danits in 1970 and Edmond Karp in 1972 guarantee the polynomial time complexity. It suggests that in each iteration, find the shortest augmenting path (path that has least number of edges) each time. Let $d(v)$ be the length of shortest $s-v$-path in G_f. Suppose we augment f to f' by a shortest $s-t$-path in G_f.

It can be proved:

Lemma 2.8 *$d_f(v)$ is monotonically increasing after each iteration.*

A critical edge on any path p is an edge whose residual capacity is equal to ϵ i.e. the edge disappears from G_f after apply the augmenting path.

Lemma 2.9 *Every edge becomes critical at most $\frac{|V|}{2} - 1$ time.*

Proof: Consider edge uv. When uv becomes critical for the first time we must have $d_f(v) = d_f(u) + 1$. Once the flow is augmented, this edge disappears from G_f

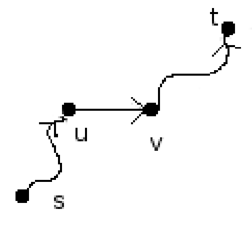

Figure 3: uv Critical Path

and it appears only if vu appears on an augmenting path and therefore the flow on uv increases. So we must have a situation (with a flow f') such that $d_{f'}(u) = d_{f'}(v) + 1$ and we know that $d_f(v) \leq d_{f'}(v)$ as it is monotonically increasing. So we will have:

$$d_{f'}(u) = d_f(v) + 1 \geq d_f(v) + 1 = d_f(u) + 2$$

So $d_f(u)$ increases by two each time after the first time. Since $d_f(u)$ is bounded by $|V| - 1$, the lemma follows. ■

According to lemma 2.9 number of times we can find augmenting path is $O(nm)$; using breadth first search we can find a shortest augmenting path in time $O(m)$. So the total time complexity will be $O(m^2n)$ which can be improved to $O(n^3)$.

2.2 Some Applications

Many problems (including some of the ones we have looked at) can be reduced to max-flow problem. We list a few below.

- Maximum Bipartite Matching: Suppose we are given a bipartite graph $G = (A \cup B, E)$ and our goal is to compute a maximum matching in G. We can create a source s and a destination t, connect s to all nodes in A with directed edges out of s with capacity 1 and connect all the nodes in B to t with edges directed to t and with capacity 1. Also direct all the edges from A to B and put capacity 1. It is easy to see that a maximum flow in this new graph corresponds to a maximum matching between A and B (i.e. those edges between A and B with non-zero flow form a matching).

- Min-Cut: A minimum cut between s and t is basically the minimum number of edges whose removal from G disconnects s from t. By using max-flow we can find a minimum $s-t$-cut. Also, the (global) minimum cut in a connected graph G is the minimum number of edges whose removal disconnects G. Clearly by computing minimum $s - t$ cut for all pairs of vertices as source and sink we can find minimum cut of G (although there are more efficient algorithms to find the min-cut).

References

S03 SCHRIJVER, ALEXANDER , Combinatorial optimization: polyhedra and efficiency, Volume 1, 2003, pp. 438–440.