Introduction to Combinatorics Lecture 16

Combinatorial Optimization

- "Optimization" plays the most important role in applications.
- If it is not of continuous type, then we refer this type of optimization problems as combinatorial optimization.
- So, we are aiming at minimizing or maximizing combinatorial objects.

Minimizing problems.

1. Chromatic number, Chromatic index of graphs
2. Domination number
3. Vertex cover, Edge cover
4. Genus, Thickness, Crossing number
5. Decycling number
6. Optical index
7. Minimum spanning tree, Traveling salesman problem (TSP)
8. Connectivity, Edge-connectivity
9. Arboricity, Linear arboricity

Maximizing problems.

1. Independence number, Maximum clique
2. Maximum matching
3. Maximum flow in networks
4. Longest cycle in graphs
5. Diameter, Longest path in graphs (hypergraphs)

6. Maximum genus
7. Hamilton cycle problem (Longest cycle, Perimeter of a graph)

Some of the above mentioned problems are solved in the sense of finding an algorithm which uses polynomial time. For example, the minimum spanning tree problem in a weighted connected graph, the maximum matching problem, the longest path problem and the maximum flow problem in a network with rational capacity.

The idea of using algorithm to solve a problem in Graph Theory is known as "Algorithmic Graph Theory". Almost all problems in Graph Theory can be "partially" solved or solved by using algorithm.

If there exists a polynomial time algorithm to find the solution, then we consider the problem is solved. Of course, we are not limited to consider special classes of graphs if we claim the problem is solved in general.

MST-problem.

One of the well-known problem that is solved is "MST-problem", i.e., finding a minimum spanning tree (total weights) in a weighted connected graph with real weights.

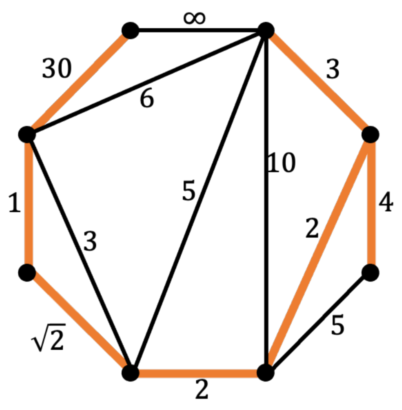

Figure 16.1

The solution can be obtained by selecting edges with minimum weight among the edges left as long as we don't create a cycle.

Maximum matching problem.

Another problem that is solved is finding a "maximum matching" of a graph.

Theorem 16.1. *Let G be a graph and M be a matching in G. Then either M is a matching of maximum cardinality or there exists an M-augmenting path.*

Proof. Clearly, if M is of maximum cardinality, then no M-augmenting paths exist. On the other hand, if M' is a matching with larger cardinality than $|M|$, let $G' = (V, M \cup M')$. Then, $\triangle(G') = 2$. This implies that the components of G' are either a path or a cycle. Since $|M'| > |M|$, at least one component of G' contains more edges from M' than that from M. Such a component is in fact an augmenting path. □

But, for more problems mentioned above, finding solutions for general graphs are very difficult. From the "algorithm" point of view, they are NP-hard, i.e., so far no polynomial time algorithms have been obtained.

Min-max problems.

A type of problems are called min-max problems. First, we review the well-known Menger's Theorem.

Theorem 16.2 (Menger, 1927)**.** *The maximum number of internally disjoint $s-t$ paths is equal to the minimum size of an $s-t$ cut (or $\langle s,t\rangle$-separating set).*

Remark.

- We can extend $s,\ t$ to $S,\ T$.
- We can extend the undirected version to directed version.
- We can also extend "vertex-disjoint paths" version to "edge-disjoint paths" version.

Idea of proof.

- The idea can be depicted as Figure 16.2.
- $c \geq n$ (in general): trivial observation.

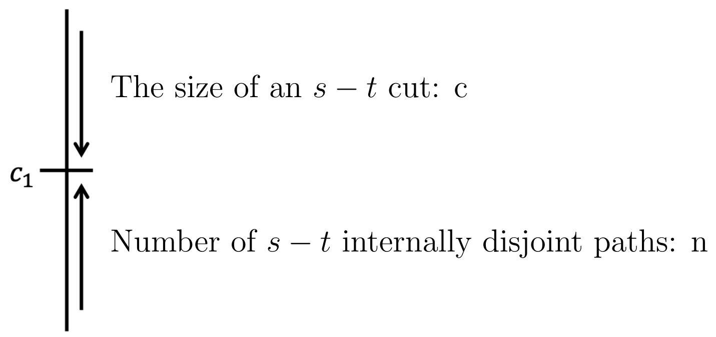
The size of an $s-t$ cut: c
c_1
Number of $s-t$ internally disjoint paths: n

Figure 16.2

- So, the main proof comes from the existence c_1 "paths" and an $s-t$ cut of size "c_1". Therefore, c_1 is the answer.
- So, we prove that if c_1 is the minimum size of an $s-t$ cut, then there exist c_1 internally disjoint $s-t$ paths.

This theorem has a beautiful extension to networks.

Definition 16.1 (Flow)**.** Let $D=(V,A)$ be a directed graph and let s (source) and t (sink) be two vertices in V. A function $f: A \to \mathbb{R}$ is called an $s-t$ flow if

1. $\forall a \in A,\ f(a) \geq 0$, and
2. $\forall v \in V \setminus \{s,t\},\ \sum_{x \in \delta^{in}(v)} f(x) = \sum_{y \in \delta^{out}(v)} f(y)$, where x has head v and y has tail v.

Definition 16.2 (Value of a flow)**.** The value of f, $value(f) =_{def} \sum_{x \in \delta^{out}(s)} f(x) - \sum_{y \in \delta^{in}(s)} f(y)$.
Equivalently, $value(f) = \sum_{x \in \delta^{in}(t)} f(x) - \sum_{y \in \delta^{out}(t)} f(y)$.
($\delta^{in}(U) = \bigcup_{u \in U} \delta^{in}(u)$, $\delta^{out}(U) = \bigcup_{u \in U} \delta^{out}(u)$, and $\delta^{in}(u)$ (resp. $\delta^{out}(u)$) is the set of arcs in A with head u (resp. tail u).)

Definition 16.3 (Capacity)**.** In a network $D=(V,A)$, a capacity function is a mapping $c: A \to \mathbb{R}_+$. We say that a flow f is subject to c if $f(a) \leq c(a)$ for each $a \in A$.

Definition 16.4 (Cut)**.** A cut in a network is a subset U of V which contains s but not t, i.e., $\langle U, V \setminus U\rangle$.

Definition 16.5 (Capacity of a cut)**.** The capacity of a cut U is defined as

$$c(U) =_{def} c(\delta^{out}(U) = \sum_{a \in \delta^{out}(U)} c(a)$$

where $\delta^{out}(U)$ denotes the set of arcs (x, y) with $x \in U$ and $y \in V\setminus$ (leaving U), and $\delta^{in}(U)$ denotes the set of arcs (x, y) with $x \in V \setminus U$ and $y \in U$ (entering U).

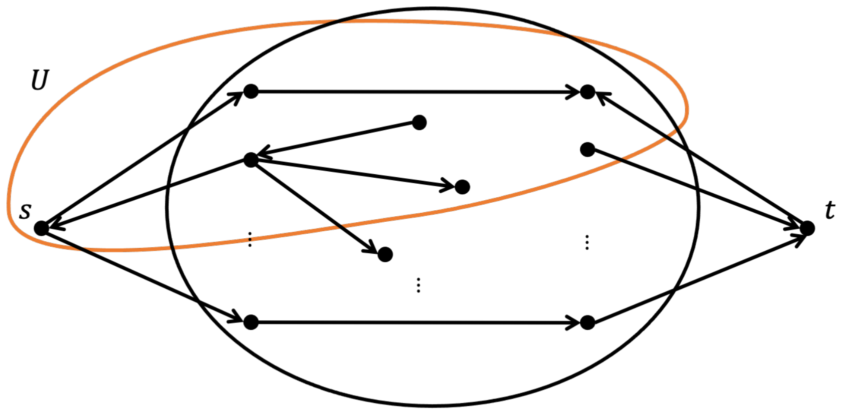

Figure 16.3

Proposition 16.3. *In a network $D = (V, A)$, $value(f) \leq c(\delta^{out}(U)) = c(U)$, where U is a cut and f is a flow from s to t.*

Proof.

$$\begin{aligned}
value(f) &= \Big(\sum_{a\in\delta^{out}(s)} f(a) - \sum_{a\in\delta^{in}(s)} f(a)\Big) + \sum_{v\in U\setminus s}\Big(\sum_{a\in\delta^{out}(v)} f(a) - \sum_{a\in\delta^{in}(v)} f(a)\Big)\\
&= \Big(\sum_{a\in\delta^{out}(s)} f(a) - \sum_{a\in\delta^{in}(s)} f(a)\Big) + \sum_{v\in U\setminus s} 0\\
&= \sum_{a\in\delta^{out}(U)} f(a) - \sum_{a\in\delta^{in}(U)} f(a)\\
&\leq \sum_{a\in\delta^{out}(U)} f(a) \leq \sum_{a\in\delta^{out}(U)} c(a) = c(\delta^{out}(U)) = c(U).
\end{aligned}$$

□

Remark. The equality $value(f) = c(U)$ holds if (1) $\forall a \in \delta^{in}(U),\ f(a) = 0$ and (2) $\forall a \in \delta^{out}(U),\ f(a) = c(a)$.

Theorem 16.4 (max-flow min-cut theorem)**.** *For any network $D = (V, A)$ with source s and sink t, the maximum flow value f from s to t is equal to the minimum capacity $c : A \to \mathbb{N}$ of an $s - t$ cut.*

(We consider integral capacity in this theorem. For the case when the capacity is either rational or real can be obtained by more careful arguments.)

Proof. By the edge-version of Menger's theorem, the maximum number of edge-disjoint $s - t$ paths is equal to the minimum size of an $s - t$ edge-cut. Note that this theorem is true for multi-digraph as well.
Now, we define a new digraph D' by letting each arc $a \in A$ be replaced by $c(a)$ arcs (with the same orientation). Thus, we have a directed multi-graph. By Menger's theorem, in D', the maximum number of $s-t$ edge-disjoint directed paths is equal to an $s-t$ edge-cut E' with minimum size.
Therefore, $D' - E'$ contains no dipaths from s to t. This implies that we have an $s - t$ edge-cut in D with capacity $|E'|$ and also a cut U in D satisfying $\sum_{a\in\delta^{out}(U)} c(a) = |E'|$.
Clearly, this is the maximum number of $s - t$ dipaths by Menger theorem. □

Remark. U can be obtained by using the arc-induced subgraph of D' by E'.

<u>Finding a maximum flow.</u> (Flow augmenting algorithm)

Idea.

1. First, we start with an "initial flow f" from s to t, say $value(f) = 0$.
2. Then, from s to t we have an $s - t$ path (directed) $P = \langle s = v_0, v_1, v_2, ..., v_k = t\rangle$ where $a_i = (v_{i-1}, v_i),\ i = 1, 2, ..., k$.
3. P is called a flow augmenting path if for each $i = 1, 2, ..., k$, either $a_i \in A,\ \sigma_i = c(a_i) - f(a_i) > 0$ or $a_i^{-1} \in A,\ \sigma_i = f(a_i^{-1}) > 0$.

   (Note that for the initial flow, the second case won't happen.)

4. If for each $i = 1, 2, ..., k$, $\sigma_i > 0$, then we can increase the current flow value by $\sigma = \min\{\sigma_1, \sigma_2, ..., \sigma_k\}$. Define a new flow f'.

$$f'(a) = \begin{cases} f(a) + \sigma & \text{if } a = a_i \text{ and } \sigma_i = c(a_i) - f(a_i) > 0; \\ f(a) - \sigma & \text{if } a = a_i^{-1} \text{ and } \sigma_i = f(a_i^{-1}) > 0; \\ f(a) & \text{if } a \notin P. \end{cases}$$

- If there are no flow augmenting paths left, then the flow value is maximum.

The above algorithm was obtained by Ford and Fulberson long time ago and it is known as "maximum flow algorithm" now. There are applications in network by using the above theorem, especially on transportation and networking. Here, we present applications in proving the other theorem in Combinatorics.

Example 1. (Hall's marriage theorem)

Let $G = (X, Y)$ be a bipartite graph. If for each subset $A \subseteq X$, $|N_G(A)| \geq |A|$, then G has a matching saturates A.

Proof. Define a network as follows: Let $D = (V, A)$ be the network where $V = \{s, t\} \cup X \cup Y$, $A = A_1 \cup A_2 \cup A_3 = \{(s, x) \mid x \in X\} \cup \{(y, t) \mid u \in Y\} \cup \{(x, y) \mid xy \text{ is an edge of } G\}$, each arc of $A_1 \cup A_2$ has capacity 1 and each arc of A_3 has capacity $M > |X|$.
Now, we claim that the network contains a flow with value $|X|$. This implies the sufficiency of Hall's theorem. First, let U be a cut with minimum capacity, see Figure 16.4. $U = \{s\} \cup X_1 \cup Y_1$. Then, the capacity of cut $c(U) = |X| - |X_1| + |Y_1|$. Since $c(U)$ is minimum, $N_D(X_1) \subseteq U_1$. By assumption, $|Y_1| \geq N_D(X_1) \geq |X_1|$. Hence, $c(U) \geq |X|$. This concludes the proof. $\square$

Example 2.

For a bipartite graph $G = (X, Y)$, define the deficiency

$$def(A) =_{def} \begin{cases} |A| - |N_G(A)| & \text{if } |A| > |N_G(A)| \\ 0 & \text{otherwise.} \end{cases}$$

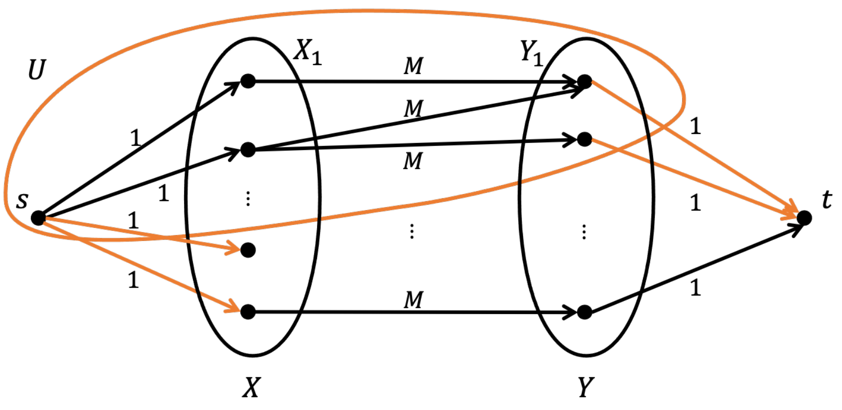

Figure 16.4

Then, $def(G) =_{def} \{def(A) \mid A \subseteq X\}$. The revised version of Hall's marriage theorem is to prove that G contains a matching of size $|X| - def(G)$.

Proof. By using max-flow min-cut theorem, we are able to prove the revised version. Mainly, using the same idea as Example 1 and we are able to find a maximum flow with value $|X| - def(G)$. (Hall's condition shows that $def(G) = 0$.) □

Example 3.

A $(0, 1)$-matrix is double stochastic if its row sums and column sums are constant.
e.g.

$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

A double stochastic matrix with row sums and column sums 2.

Prove that a double stochastic matrix can be written as the sum of permutation matrices.

Proof. Let $M = \left[m_{i,j}\right]_{n\times n}$ be the matrix we consider, furthermore, let its row sum (resp. column sum) be k. Clearly, $1 \le k \le n$ and M can be corresponded to a bipartite graph (A, B) where $|A| = |B| = n$. Similar to Example 1, we can define a network with source s and sink t. Also, the capacities of arcs are defined by the same way.
Now, it is suffices to prove that the minimum cut must be of capacity n (?) and thus we obtain a flow with value n. This shows that M can be the sum of a permutation matrix

and a double stochastic matrix M' with row sum and column sum $k-1$. Hence, the proof follows by induction on k. $\square$

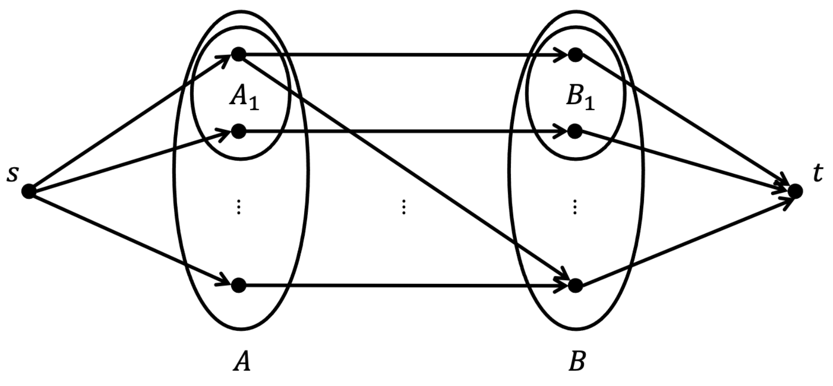

Figure 16.5

Remark. (the (?) part) Let $U = \{s\} \cup A_1 \cup B_1$ be the minimum cut. Since $M > n$, $N(A_1) \subseteq B_1$. By assumption, the degree sum of vertices in A_1 (in (A, B)) is equal to $|A_1| \cdot k$. This implies that the degree sum of vertices of B_1 (in (A, B)) is at least $|A_1| \cdot k$. But, each vertex of B_1 is of degree k (in (A, B)), thus $|B_1| \geq |A_1|$. Now, $c(U) = |A| - |A_1| + |B_1| \geq |A| = n$.

Example 4.

A function $f : A \to \mathbb{R}$ defined on a directed graph $D = (V, A)$ is called a circulation if for each vertex $v \in V$, we have

$$\sum_{a \in \delta^{in}(v)} f(a) = \sum_{b \in \delta^{out}(v)} f(b).$$

Note that the flow conservation law holds at each vertex v.

The following theorem can be proved by using max-flow min-cut theorem, we omit the details here.

Theorem 16.5 (Hoffman, 1960)**.** *Let $D = (V, A)$ and $d, c : A \to \mathbb{R}$ satisfying $d(a) \leq c(a)$ for each $a \in A$. Then, there exists a circulation f such that $d(a) \leq f(a) \leq c(a)$ $\forall a \in A$ if and only if for each subset $U \subseteq V$, $\sum\limits_{a \in \delta^{in}(U)} d(a) \leq \sum\limits_{a \in \delta^{out}(U)} c(a)$, i.e., $d(\delta^{in}(U)) \leq c(\delta^{out}(U))$.*

Final words.

Combinatorics or Combinatorial Theory play an important role in modern era especially on discrete models. This one semester course can only provide some parts of the topic due to the limit of time. In fact, I am not sure that we are able to cover everything even we are given infinite amount of time. At the time about to finish, there are new topics to occur. Always, new ideas to come and thus new topics to learn. So, we just have to move toward as long as we learn "Combinatorics", the world of counting.