Lecture Notes: Edmond's Blossom Algorithm

Palash Dey
palash.dey@cse.iitkgp.ac.in
Indian Institute of Technology Kharagpur

August 15, 2023

Given an unweighted graph $G(V, E)$, the maximum cardinality matching problem asks to compute a matching in G of maximum cardinality. Let us begin with working towards an upper bound on the size (number of edges) of a maximum cardinality matching. Let $U \subseteq V$ be any arbitrary subset of V. We denote the number of components of odd cardinality in $G[V \setminus U]$ by $o(V \setminus U)$. We observe that any component of odd cardinality cannot be perfectly matched within itself (that is, using only the edges whose both endpoints are in that component). Any such odd component C needs at least one partner outside C and such a partner can come only from U (why?). Hence, every matching of G necessarily leaves at least $o(V \setminus U) - |U|$ vertices of G unmatched. Let M be any matching of maximum cardinality; size of the matching M, denoted by $|M|$, is the number of edges in it. Then we have

$$|M| \leqslant \frac{1}{2}(|V| - (o(V \setminus U) - |U|)) = \frac{1}{2}(|V| + |U| - o(V \setminus U)).$$

Since the above inequality holds for every $U \subseteq V$, we have

$$|M| \leqslant \min_{U \subseteq V} \frac{1}{2}(|V| + |U| - o(V \setminus U)).$$

The celebrated Tutte-Berge Theorem says that the above inequality is actually tight.

Theorem 1 (Tutte-Berge Theorem) *For any maximum cardinality matching M of an undirected graph G, we have*

$$|M| \leqslant \min_{U \subseteq V} \frac{1}{2}(|V| + |U| - o(V \setminus U)).$$

We now prove this result by exhibiting an algorithm thanks to Edmond. It starts with empty matching and iteratively finds another matching of size one more until we have a matching of maximum cardinality. To finish the description of the algorithm, we need to answer the following questions.

1. Given a matching M which is not of maximum cardinality, how do we compute another matching M' such that $|M'| > |M|$?
2. Given a matching M, how to certify that M is a matching of maximum cardinality?

Let us begin with the first question. The new idea here is M-augmenting path. A path $P = (x_0, x_1, \ldots, x_k)$ is an M-augmenting path if (i) x_0 and x_k are unmatched (a.k.a free, exposed, etc.) vertices and (ii) $\{x_{2i-1}, x_{2i}\} \in M$ for every $1 \leqslant i \leqslant k/2$; that is edges alternate from matching and out of matching, beginning and ending with unmatched vertices. What is the big

deal if we find an $\mathcal{M}$-augmenting path $\mathcal{P}$? We can obtain a matching $\mathcal{M}'$ with $|\mathcal{M}'| = |\mathcal{M}| + 1$: $\mathcal{M}' = \mathcal{M}\Delta\mathcal{P}$[1]. Hence, if we have found an $\mathcal{M}$-augmenting path, we can use it to obtain another matching of size one more than the size of $\mathcal{M}$. Does an $\mathcal{M}$-augmenting path guaranteed to exist if $\mathcal{M}$ is not a matching of maximum cardinality? The answer is yes! To see this, let $\mathcal{M}$ be any matching of not maximum cardinality and $\mathcal{M}'$ be another matching such that $|\mathcal{M}'| > |\mathcal{M}|$. Consider the subgraph $\mathcal{H} = \mathcal{M}\Delta\mathcal{M}'$; it is a collection of alternating cycles and alternating paths (where the edges alternate between $\mathcal{M}$ and $\mathcal{M}'$). In alternating cycles, we have the same number of edges from $\mathcal{M}$ and $\mathcal{M}'$. Among alternating paths, only $\mathcal{M}$-augmenting paths have more edges from $\mathcal{M}'$ than $\mathcal{M}$. Hence, if we do not have any $\mathcal{M}$-augmenting path in $\mathcal{H}$, then we have $|\mathcal{M}| \leqslant |\mathcal{M}'|$ (why?) contradicting our assumption that $|\mathcal{M}'| > |\mathcal{M}|$. Hence, an $\mathcal{M}$-augmenting path is guaranteed to exist if $\mathcal{M}$ is not a maximum cardinality matching. Moreover, it is computationally easy to obtain $\mathcal{M}'$ from $\mathcal{M}$ and an $\mathcal{M}$-augmenting path $\mathcal{P}$: $\mathcal{M}' = \mathcal{M}\Delta\mathcal{P}$. This finishes our answer to the first question. This is also known as Berge's Theorem.

Theorem 2 (Berge's Theorem) *A matching $\mathcal{M}$ is a maximum cardinality matching in a graph $\mathcal{G}$ if and only if there is no $\mathcal{M}$-augmenting path in $\mathcal{G}$.*

From algorithmic perspective, all we need is a recipe which is guaranteed to find an $\mathcal{M}$-augmenting path when one exists (which is exactly when $\mathcal{M}$ is not a matching of maximum cardinality).

Algorithm for Finding an $\mathcal{M}$-Augmenting Path

Since we are looking for an $\mathcal{M}$-augmenting path, it makes sense to start our search from an unmatched vertex since every $\mathcal{M}$-augmenting path starts at some unmatched vertex. To make our search systematic, we perform the following modified version of breadth first search (BFS). We begin with marking all unmatched vertices "even" (denoting the fact that these vertices are at even, namely zero, level in the BFS forest that we are building) and enqueueing them like standard BFS; other vertices are unlabeled. An important difference between our modified BFS and standard BFS is that we enqueue only vertices marked "even" which are precisely the vertices at an even distance from the root of some tree in the forest that we are maintaining. In every iteration, we first dequeue a (marked "even") vertex, say u. Let $\{u, v\}$ be an edge. We have the following possibilities:

1. (easy) If the vertex v is marked "odd," then we simply ignore this edge. That is, we simply ignore edges between an even vertex and another vertex which is already marked "odd."

2. (easy) If the vertex v is unmarked, we observe that v must be a matched vertex; let $\mathcal{M}(v)$ be its mate (matched vertex). In this case, we add the edges $\{u, v\}$ and $\{v, \mathcal{M}(v)\}$ in the forest and mark v and $\mathcal{M}(v)$ "odd" and "even" respectively. Hence, every vertex which is marked "odd" has exactly one child in the forest we are maintaining.

3. (easy) If the vertex v is marked "even" and belongs to some other tree (that is, not the tree where u belongs) in the forest that we are maintaining, then we have obtaining an $\mathcal{M}$-augmenting path — root of u's tree $\dashrightarrow u \to v \dashrightarrow$ root of v's tree. Hence, we are done in this case; remember our goal is to find an $\mathcal{M}$-augmenting path if one exists.

4. (pure genius) If the vertex v is marked "even" and belongs to the same tree as u. In this case, we have obtain a structure like Figure 1 which is called a flower. A flower has a cycle

[1] We slightly abuse the notation $\mathcal{P}$ to also denote the set of edges in the path $\mathcal{P}$. For two sets $\mathcal{X}$ and $\mathcal{Y}$, we define $\mathcal{X}\Delta\mathcal{Y} = (\mathcal{X} \setminus \mathcal{Y}) \cup (\mathcal{Y} \setminus \mathcal{X})$; this is also known as symmetric difference between $\mathcal{X}$ and $\mathcal{Y}$.

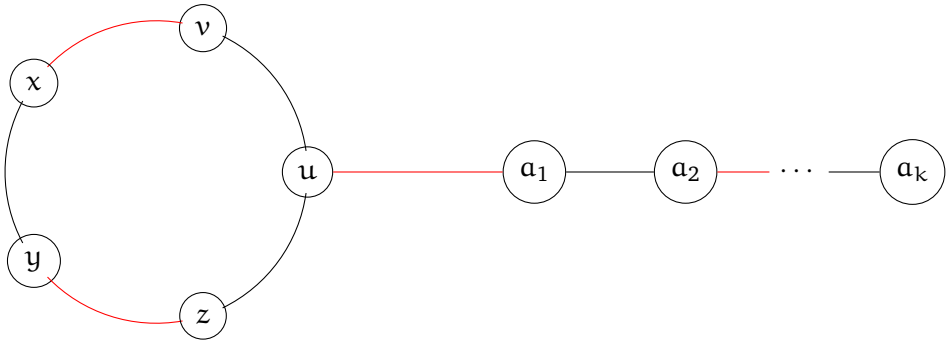

Figure 1: Schematic diagram of a flower. Red edges are matching edges. The cycle is called blossom and the path between u and a_k is called stem.

of odd length called a blossom and an even length alternating path called stem from an unmatched vertex a_k. In our modified BFS, the root of the tree of u serves the role of a_k which is an unmatched vertex. Note that, the stem could be empty but not the blossom. If the flower obtained has no stem, then we are good, otherwise we remove the matched edges in the stem from $\mathcal{M}$ and add the remaining edges in the stem in $\mathcal{M}$ thereby obtaining another matching of the same size as $\mathcal{M}$ but we now have a flower with empty stem.

Let $\mathcal{B}$ be that flower which is just a blossom; $\mathcal{B}$ has exactly one unmatched edge. How do we proceed when we have found a blossom $\mathcal{B}$ (mentioning for the last time that $\mathcal{B}$ does not have any stem)? We collapse the vertices of $\mathcal{B}$ into one (super)node, mark that (super)node unmatched, and restart our task of finding an $\mathcal{M}$ augmenting path in the resulting graph. The collapsing operation is formally called contracting the vertex set $\mathcal{B}$ and resultant graph is denoted by $\mathcal{G}/\mathcal{B}$.

Proof of Correctness

We next show that there is an $\mathcal{M}$-augmenting path in $\mathcal{G}$ if and only if there is an $(\mathcal{M}/\mathcal{B})$-augmenting path in $\mathcal{G}/\mathcal{B}$. In one direction, let a path $\mathcal{P}$ between x and y be an $\mathcal{M}$-augmenting path in $\mathcal{G}$. Since both x and y are unmatched, let us assume without loss of generality that $x \neq u$. Let z be the first vertex in $\mathcal{P}$ from x which belongs to $\mathcal{B}$; if no such z exists, then $\mathcal{P}$ is an $(\mathcal{M}/\mathcal{B})$-augmenting path in $\mathcal{G}/\mathcal{B}$. We observe that the prefix of the path $\mathcal{P}$ from x to z is an $(\mathcal{M}/\mathcal{B})$-augmenting path in $\mathcal{G}/\mathcal{B}$ since z becomes the supernode in $\mathcal{G}/\mathcal{B}$ which is unmatched in $\mathcal{G}/\mathcal{B}$.

In the other direction, let $\mathcal{Q}$ be an $(\mathcal{M}/\mathcal{B})$-augmenting path in $\mathcal{G}/\mathcal{B}$. If $\mathcal{Q}$ does not pass through the supernode, then $\mathcal{Q}$ is an $\mathcal{M}$-augmenting path in $\mathcal{G}$ too. So let us assume that $\mathcal{Q}$ passes through the supernode. Since the supernode is unmatched in $\mathcal{G}/\mathcal{B}$, it must be one of the end vertices in $\mathcal{Q}$. Let z_1 be the neighbor of the supernode in $\mathcal{Q}$ and z_2 the neighbor of z_1 in $\mathcal{B}$ in $\mathcal{G}$; observe that $\{z_1, z_2\} \notin \mathcal{M}$ since the supernode is unmatched in $\mathcal{G}/\mathcal{B}$. If z_2 is u which is an unmatched vertex in $\mathcal{G}$ under $\mathcal{M}$, then replacing the supernode with u gives us an $\mathcal{M}$-augmenting path in $\mathcal{G}$. Otherwise, we replace the supernode in $\mathcal{Q}$ with z_2 and concatenate the $\mathcal{M}$-alternating from z_2 to u of even length in $\mathcal{B}$ to obtain an $\mathcal{M}$-alternating path in $\mathcal{G}$.

An important point to note that an $\mathcal{M}$-augmenting path in $\mathcal{G}$ can be computed from an $(\mathcal{M}/\mathcal{B})$-augmenting path in $\mathcal{G}/\mathcal{B}$ in $\mathcal{O}(m)$ time. Recall that we have already reduced our problem of computing a matching of maximum cardinality to the problem of computing an $\mathcal{M}$-augmenting path from a given matching $\mathcal{M}$.

What remains to show to prove the correctness of the algorithm is that when none of the four

steps are applicable, the current matching $\mathcal{M}$ is indeed a matching of maximum cardinality. We observe that when none of the four steps are applicable, all the edges not present in the forest must have one end point marked "even" and the other end point marked "odd." Moreover, all the forest edges are also between "even" and "odd" vertices. Let us define $\mathcal{U}$ to be the set of vertices marked "odd." Then all the vertices marked "even" becomes isolated in $\mathcal{G}[\mathcal{V} \setminus \mathcal{U}]$; that is $o(\mathcal{V} \setminus \mathcal{U}) = |\text{Even}|$ where Even is the set of vertices marked "even." On the other hand,

$$|\mathcal{M}| = |\text{Odd}| + \frac{1}{2}(|\mathcal{V}| - |\text{Even}| - |\text{Odd}|) = \frac{1}{2}(|\mathcal{V}| + |\text{Odd}| - |\text{Even}|) = \frac{1}{2}(|\mathcal{V}| + |\mathcal{U}| - o(\mathcal{V} \setminus \mathcal{U})).$$

Notice that, $\frac{1}{2}(|\mathcal{V}| + |\mathcal{U}| - o(\mathcal{V} \setminus \mathcal{U}))$ is an upper bound on the cardinality of any matching in $\mathcal{G}$. Hence, $\mathcal{M}$ is a matching of maximum cardinality. Hence, when the algorithm does not find any augmenting path or flower, then the current matching is indeed a matching of maximum cardinality. This concludes the proof of correctness of Edmond's algorithm. This also proves Tutte-Berge Theorem.

Runtime of Blossom Algorithm

Finally, we analyze the runtime of this algorithm. Since the maximum cardinality matching can have at most $\frac{n}{2}$ edges, we can augment the current matching at most $\frac{n}{2}$ times. Between two consecutive augmentations, how many times we can find a blossom (and thus shrink it and restart our modified BFS)? At most $\frac{n}{2}$ times since contracting a blossom reduces the number of vertices by at least two. Each run of our modified BFS can be implemented in $\mathcal{O}(m)$ time. Hence, the overall runtime of our algorithm is $\mathcal{O}(mn^2)$.