Matchings in Graphs

Lecturer: *Meena Mahajan*
Scribe: *Rajesh Chitnis*

Meeting: 1
6th Jan 2010

Most of the material in this lecture is taken from the book "Fast Parallel Algorithms for Graph Matching Problems" by Karpinski and Rytter, [KR98].

We will only be considering simple undirected finite graphs unless stated otherwise. Graphs will be denoted as $G = (V, E)$.

1 Some preliminary definitions

Definition 1 *Let $G = (V, E)$ be a graph. $M \subseteq E$ is called as a **matching** of G if $\forall\ v \in V$ we have $\mid \{e \in M : v \text{ is incident on } e \in E\} \mid \leq 1$.*

Definition 2 *A matching M of G is said to be **maximal** if $\forall\ e \in E \setminus M$ the set of edges given by $M \cup \{e\}$ is not a matching of G.*

Definition 3 *The **size** of a matching M of G is the number of the edges it contains and is denoted by $\mid M \mid$.*

Definition 4 *A matching M of G is said to be **maximum** if $\forall$ matching M' of G we have $\mid M \mid \geq \mid M' \mid$. A maximum matching is always maximal but not vice-versa.*

Definition 5 *Let M be matching of G. A vertex $v \in V$ is said to be **M-saturated** if M contains an edge incident on v. Otherwise v is said to be **M-unsaturated** or **M-free**.*

Definition 6 *A matching M of G is said to be **perfect** if all vertices of G are M-saturated. A graph with an odd number of vertices can never admit a perfect matching.*

Definition 7 *A matching M of G is said to be **near-perfect** if exactly one vertex of G is M-unsaturated. A graph with an even number of vertices can never admit a near-perfect matching.*

Definition 8 *Let $A \subseteq V$. A matching M of G is said to be **A-perfect** if each vertex in A is M-saturated. A perfect matching is a V-perfect matching.*

2 Some Remarks

Remark 9 *We will later look at weighted graphs i.e graphs with a weight function* $w : E \rightarrow \mathbb{R}^+ \cup \{0\}$. *There we will be interested in finding matchings of maximum weight where the weight of a matching is the sum of weights of edges which are in the matching.*

Remark 10 *We will consider three types of problems*

- *Decision - Does* G *have a matching of size* $\geq k$ *?*
- *Search - Find a matching in* G *of size* $\geq k$
- *Counting - How many matchings of* G *have size* $\geq k$*?*

3 Augmenting Paths

Definition 11 *A path* P *in* G *is said to be* ***M-alternating*** *if the edges of* P *alternate with respect to membership in* M.

Definition 12 *A path* P *in* G *is said to be* ***M-augmenting*** *if* P *is a maximal* M*-alternating path starting and ending at vertices which are* M*-unsaturated. Clearly, every* M*-augmenting path must have odd length (number of egdes).*

Lemma 13 *Let* G *be a graph whose maximum degree is atmost 2. Then every component of* G *is either an isolated point, a path or a cycle.*

Proof: Consider any non-isolated vertex v of G. Its (at most two) neighbours further have degree at most 2 and so on. So the component of G containing v is either a path or a cycle. This holds true for all non-isolated vertices of G and hence we are done. ■

Lemma 14 *(Berge 1957) A matching* M *is maximum if and only if* G *has no* M*-augmenting path.*

Proof: Suppose there exists an M-augmenting path P. Consider the symmetric difference $M \oplus P$ (edges which are present in exactly one of M or P). Since P is an M-augmenting path, $M \oplus P$ is also a matching of G and $|M \oplus P| = |M| + 1$. So M is not maximum.

Suppose M is not maximum. Let M' be a maximum matching and so we have $|M'| > |M|$. Consider $M \oplus M'$. Each vertex has degree at most 2 in $M \oplus M'$ as each of M and M' can contribute atmost 1 each to degree of each vertex in $M \oplus M'$. By Lemma 13, $M \oplus M'$ consists of cycles and paths and isolated vertices. But edges of $M \oplus M'$ are alternate in belonging exclusively to M and M'. Hence each cycle must be even. So M' can score over M in size only from the paths. So, there exists at least one path in $M \oplus M'$ which has more edges from M' than from M. But such a path is M-augmenting. ■

Corollary 15 *(Hopcroft-Karp) Let M^* be a matching of G. Then for any matching M of G such that $|M^*| \geq |M|$, we have $|M^*| - |M|$ vertex-disjoint M-augmenting paths. The non-M edges on these paths all belong to M^*.*

Proof: Refer to proof of Lemma 14. Every cycle of $M \oplus M^*$ is even and every path of $M \oplus M^*$ which is not M-augmenting must have equal number of edges from M and M^* as M^* is maximum. Also note that each M-augmenting path has exactly one edge more from M^* than from M. So we need $|M^*| - |M|$ such paths; these are all vertex-disjoint as we defined augmenting paths as **maximal** paths starting and ending at unsaturated points (see Definition 12). ■

Corollary 16 *Let M^* be a maximum matching and M be any matching. If M is not maximum, then the shortest M-augmenting path has length $\leq \frac{|V|}{|M^*|-|M|} - 1$*

Proof: From Corollary 15 we know that there are $|M^*| - |M|$ vertex-disjoint (and hence edge-disjoint) M-augmenting paths. By Pigeonhole Principle, one of the paths must have at most $\frac{|V|}{|M^*|-|M|}$ vertices and thus has length at most $\leq \frac{|V|}{|M^*|-|M|} - 1$. ■

4 Algorithm for finding maximum matching using augmenting paths

Consider the following algorithm; its correctness follows immediately from Lemma 14.

1. $M = \emptyset$
2. **while** there is an M-augmenting path P
   do $M \leftarrow M \oplus P$
3. **return** M

The challenge now is to detect existence of and find augmenting paths efficiently. We will first consider the case when G is bipartite.

5 An O(n^3) algorithm for finding maximum matching in bipartite graphs

Let $G = (A \cup B, E)$ be a bipartite graph and let M be a matching of G. We want to find a maximum matching of G. Denote by A_0, B_0 the sets of M-unsaturated vertices in A, B respectively. We consider a new directed graph H on the vertex set $A \cup B$ and edge set E. Edges which are in M are directed $A \rightarrow B$ and edges not in M are directed $B \rightarrow A$.

Claim 17 *G has a M-augmenting path if and only if H has a path from B_0 to A_0.*

Proof: Suppose G has an M-augmenting path say from $u \in A_0$ to $v \in B_0$. The same path directed from v to u is clearly a path in H from B_0 to A_0.

Suppose H has a path from $b \in B_0$ to $a \in A_0$. The underlying undirected path from a to b is clearly an M-augmenting path. ■

So we do a depth-first-search (DFS) from B_0 and stop as soon as we reach some vertex in A_0, thus giving us an M-augmenting path P. We then augment M along P, and repeat the same process with the new matching $M \oplus P$. If we cannot reach any vertex of A_0, then we can conclude from Claim 17 that G has no M-augmenting path i.e. M is maximum.

Let us now analyse the time complexity of our algorithm. Denote $|V| = n$ and $|E| = m$.

1. Assume $|B| \leq |A|$ as otherwise we could have just swapped the roles of A and B in our algorithm. Thus $|B_0| \leq |B| \leq \frac{n}{2}$. Also at each stage of our algorithm, by augmenting, we saturate a previously-unsaturated vertex from B without doing anything to the vertices which are already saturated. So we need at most $|B_0| \leq \frac{n}{2}$ stages.

2. At each stage we may need to do several DFS, one starting from each vertex in B_0. The maximum number of times we need to do DFS is $|B_0|$, as in the worst-case, only the last vertex from B_0 that we apply DFS to may lead to a path in A_0. Recollect that a single DFS can be done in $O(n+m)$ time.

3. Once an augmenting path is found, we can augment the matching easily in $O(m)$ time.

Thus the Total Time taken by algorithm is at most $\frac{n}{2}\Big[O(m) + |B_0| * O(n+m)\Big]$. However now we use a trick to shave off the $|B_0|$ factor. Add a super-vertex β and draw edges directed from β to every point in B_0. Thus we need to apply DFS only once, for vertex β. Thus the time complexity becomes $O\Big(\frac{n}{2}[m + (n+m)]\Big) = O\Big(n^2 + nm\Big)$. Since a bipartite graph on n vertices can contain at most $(\frac{n^2}{4})$ edges, the time complexity of our algorithm is $O(n^3)$.

6 Hopcroft-Karp Algorithm for finding a maximum matching in bipartite graphs in $O(n^{2.5})$ time

In the preceding algorithm, we looked for a single augmenting path at a time and augmented it. Instead we will now find a maximal family of vertex-disjoint shortest-length augmenting paths and augment all of them together in a single stage. This improvement will help us to bring the time complexity down to $O(n^{2.5})$.

Consider the following algorithm.

1. $M = \emptyset$

2. **while** (there is an M-augmenting path),
   do find a maximal family $\mathcal{F}$ of vertex-disjoint shortest M-augmenting paths);
   set $M \leftarrow M \oplus \mathcal{F}$

3. **return** M

The correctness of the algorithm follows from Lemma 14.

We now show that using a maximal family $\mathcal{F}$ of shortest augmenting paths instead of a single augmenting path significantly reduces the number of stages (Lemma 21), and also that the time per stage due to having to find such families does not increase (Lemma 22). We need some technical lemmas.

Lemma 18 *Let M be a matching of G and let P be an M-augmenting path of shortest length. Let P' be an $(M \oplus P)$-augmenting path. Then $|P'| \geq |P| + |P \cap P'|$, where $|P|$ is the number of edges in P.*

Proof: Consider $N = (M \oplus P) \oplus P'$. Then N is clearly a matching and $|N| = |M| + 2$. Thus by Corollary 15, there are 2 vertex-disjoint M-augmenting paths, say P_1 and P_2, with the non-M edges in N. That is, $P_1 \cup P_2 \subseteq M \oplus N$. Note that $M \oplus N = P \oplus P'$ and thus we have $|P \oplus P'| \geq |P_1| + |P_2|$. But P_1, P_2 are both M-augmenting paths and P is a shortest M-augmenting path. Therefore $|P \oplus P'| \geq 2|P|$. However $|P \oplus P'| = |P| + |P'| - |P \cap P'|$ and so the desired inequality follows. ■

Lemma 19 *Let $M_0 = \emptyset$, and consider the sequence $M_0, M_1, M_2, M_3, ...$ where $\forall i$, P_i is a shortest M_i-augmenting path, and $M_{i+1} = M_i \oplus P_i$. Then, for $i < j$, $|P_i| \leq |P_j|$. Further, $|P_i| = |P_j|$ implies that P_i and P_j are vertex-disjoint.*

Proof: It follows from Lemma 18 that for $i < j$, $|P_i| \leq |P_j|$.

Suppose now that for some $i < j$, P_i and P_j are not vertex-disjoint, and assume to the contrary that $|P_i| = |P_j|$. This implies that $|P_i| = |P_{i+1}| = \ldots = |P_{j-1}| = |P_j|$. Then there exist some k, l such that $i \leq k < l \leq j$ and P_k and P_l are not vertex-disjoint and further for all m between l and k we have P_m is vertex-disjoint from both P_k and P_l. Therefore P_l is an (M_k)-augmenting path and so by Lemma 18 we have $|P_l| \geq |P_k| + |P_l \cap P_k|$. However we are given that $|P_l| = |P_k|$ which implies that $|P_l \cap P_k| = 0$ i.e. P_l and P_k have no edges in common. However since P_l and P_k are not vertex-disjoint, they have a common vertex say x and then they must have in common the edge from $M_k \oplus P_k$ which is incident on x leading to a contradiction. ■

Lemma 20 *Let $\mathcal{F}$ be an inclusion-maximal family of vertex-disjoint shortest M-augmenting paths, all of length l_1. Let l_2 be the length of a shortest $(M \oplus \mathcal{F})$-augmenting path. Then $l_2 \geq l_1 + 2$.*

Proof: Let $\mathcal{F} = \{P_1, P_2, .., P_r\}$. Let P' be a shortest $(M \oplus \mathcal{F})$-augmenting path. Note that $M \oplus \mathcal{F} = (..(M \oplus P_1) \oplus P_2)..) \oplus P_r$. Suppose P' is disjoint from each element of $\mathcal{F}$. Then P' is also an M-augmenting path, but by maximality of $\mathcal{F}$, it is not a shortest augmenting path. So $l_2 > l_1$. Next, suppose that P' has a vertex in common with at least one path in $\mathcal{F}$. By Lemma 19 we have $l_2 > l_1$. Finally note that l_1, l_2 are both lengths of augmenting paths and hence must both be odd; hence $l_2 > l_1 \Longrightarrow l_2 \geq l_1 + 2$. ■

Let us look at the graph H considered at beginning of Section 5. Let $|A \cup B| = n$ and $|E| = m$. Note that Claim 17 holds.

Lemma 21 *The algorithm described at start of Section 6 makes atmost $2\sqrt{n}$ iterations*

Proof: Let M^* be a maximum matching and let M be the matching after $\sqrt{n}$ iterations. By Lemma 20, the length of the shortest M-augmenting path is at least $(2\sqrt{n} - 1) \geq \sqrt{n}$. By Corollary 16 we have $\sqrt{n} \leq$ (length of shortest M-augmenting path)$\leq \frac{n}{|M^*|-|M|}$, and so $|M^*| - |M| \leq \sqrt{n}$. From this point onwards, even if we augment just one path in each iteration, we need at most $\sqrt{n}$ more iterations, as each augmenatation increases size of matching by 1. Thus overall we need no more than $2\sqrt{n}$ iterations. iterations ■

Lemma 22 *Each iteration of the algorithm can be implemented in $O(m)$ time.*

Proof: First we will use breadth-first-search BFS to find the length k of a shortest path from B_0 to A_0. Simultaneosuly, we produce the sequence of disjoint layers $B_0 = L_0, L_1, L_2, ..., L_k \subseteq A_0$ where

- for all $0 \leq i < k$, L_i is the set of vertices at distance i from B_0, and
- L_k is the subset of A_0 which is at distance k from B_0.

To avoid multiple BFSs from each vertex in B_0, we add a super-vertex β and draw edges from it to all vertices of B_0. Start a BFS from β to get distance of β from A_0. Subtract one to get length of shortest path from B_0 to A_0. This takes $O(m)$ time.

Now consider a modified DFS which starts at a vertex $v \in B_0$ and stops as soon as it reaches a vertex say w in L_k and outputs this $v \rightarrow w$ path. Add this M-augmenting path to $\mathcal{F}$ and delete all vertices visited in the modified DFS. This is crucial: we delete not just the augmenting path but also other vertices visited in the modified DFS. Why is this reasonable? Let x be a vertex seen at some L_j in the DFS started from $v \in B_0$. If x does not lead to an M-augmenting path of length k starting at v, then x cannot be on any M-augmenting path of length k: any such path has to begin at some vertex in B_0 and it has to use i edges to reach x.

Redo the whole procedure now starting at another vertex in B_0. Continue until all vertices of B_0 are explored. This clearly gives us a maximal family of vertex-disjoint shortest-length augmenting paths.

Let m_i be the number of edges visited in the i^{th} DFS which takes $O(m_i)$ time. Noting that $m \geq \sum_i m_i$, the time taken is $O(m)$. ■

Theorem 23 *The algorithm runs in $O(n^{2.5})$ time.*

Proof: From Lemma 22 we know that each phase can be implemented in $O(m)$ time. Also from Lemma 21 we know that there are atmost $2\sqrt{n}$ iterations. Thus time taken by our algorithm is $O(\sqrt{n}) * O(m) = O(n^{2.5})$ ■

References

[KR98] Marek Karpinski and Wojciech Rytter. *Fast parallel algorithms for graph matching problems*. Oxford University Press, Oxford, 1998. Oxford Lecture Series in Mathematics and its Applications 9.