CPSC 665 : An Algorithmist's Toolkit Lecture 1 : 16 April 2015

Perfect Matchings in Regular Bipartite Graphs in $O(n \log n)$ Time

Lecturer: Sushant Sachdeva *Scribe: Rachel Lawrence*

1. Preliminaries

Let $G = (P, Q, E)$ be a d-regular bipartite such that P and Q are disjoint vertex sets and $E \subset P \times Q$. Let $n = |P|$ be the number of vertices in P, and let $m = |E|$ be the number of edges in G.

Fact 1.1. *$m = nd$ and $|P| = |Q| = n$ as a consequence of regularity.*

Definition 1.2 (Matching). *A **matching** in graph G is a set of edges in G such that no two edges share a common vertex. A matching is **perfect** if it consists of exactly n edges. Otherwise, it is **partial**.*

Fact 1.3. *G can be decomposed into exactly d perfect matchings. This is a direct consequence of Hall's marriage theorem [Bol98] which I will not prove here, since it is included in many basic Algorithms courses).*

Definition 1.4 (Augmenting path). *Given a matching M on graph G, an **augmenting path** is a path in G such that the start and endpoints of the path are unmatched vertices, and the path alternately contains edges that are and are not in the partial matching.*

Definition 1.5 (Symmetric difference). *The **symmetric difference** of two edge sets P and M, denoted $P \Delta M$, is defined as $(P - M) \cup (M - P)$.*

Fact 1.6. *The symmetric difference of a partial matching M with augmenting path P gives a new matching $M*$ with exactly one more edge than M.*

2. Background

The standard algorithm for general (not necessarily regular) bipartite graphs is $O(m\sqrt{n})$ [HK73]. For regular bipartite graphs, deterministic algorithms exist showing that perfect matchings are computable in $O(m)$ time, and that if the graph is d-regular, this bound can be improved to $O(min\{m, \frac{n^2}{d}\})$ [GKK10]
Our focus will be on a randomized algorithm by Goel, Kapralov, and Khanna [GKK09] which finds a perfect matching in a d-regular graph in $O(n \log n)$ time, both in expectation and with high probability. Note that just outputting a matching takes $O(n)$ time, so this is intuitively quite a good bound!

After describing and proving the algorithm and its runtime, we will also see that there exists an $\Omega(nd)$ lower bound on finding perfect matchings in $d-$regular bipartitie graphs using deterministic algorithms; in other words, that randomization is the only way to achieve a bound as low as $O(n \log n)$ [GKK09].

3. The Randomized Matching Algorithm

Algorithm Sketch: The algorithm will find successive augmentations for a matching, by taking random walks on a modified "matching graph" that encodes the current partial matching.

Definition 3.1 (Matching graph)**.** *For a partial matching M in G that leaves $2k$ vertices unmatched, the **matching graph** H is created from G by the following procedure:*

(1) Orient the edges of G from P to Q
(2) Contract each pair of vertices $(u, v) \in M$ into a single "supernode"
(3) Add source node s connected by d parallel edges to each unmatched vertex in P; orient all such edges away from s
(4) Add sink node t connected by d parallel edges to each unmatched vertex in Q; orient all such edges towards t

Fact 3.2. *The following equations can be verified as a simple exercise.*

- # nodes in $H = n + k + 2$
- # edges in $H = n(d-1) + k(2d+1)$
- For all vertices $v \in H$ such that $v \neq s$ and $v \neq t$, the in-degree of v = the out-degree of v.

Lemma 3.3. *Any path from s to t in H gives an augmenting path in G with respect to M.*

Proof. First, we check that the path starts and ends on an unmatched vertex: this is clearly true, because from the construction of H, the only edges accessible from s are unmatched vertices in P, and similarly, the only edges from which t is accessible are unmatched vertices in Q.

Next, consider whether the path alternates edges that are and are not in the matching. First, note that any edge that is in the matching has been condensed into a supernode in H; therefore, we ask instead whether it is possible to visit two supernodes in a row. It is not, because to visit two supernodes along one path requires an edge between them, and this edge shares a vertex with an edge of the matching, so it cannot itself be a part of the same matching.

Furthermore, is it possible to visit two non-supernode vertices in a row? The answer is again no, because of all the edges in H, vertices in P only have edges directed into Q, and vertices in Q only have vertices directed to t. Therefore, it is impossible to create a path that travels between non-supernode vertices in P and Q more than once. This completes the proof. ☐

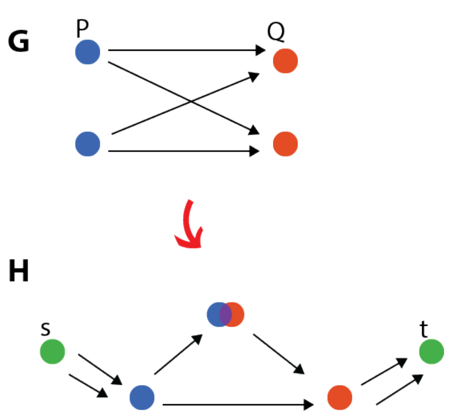

Figure 1. An example Matching Graph

We next examine a proposition that will guide the random walk portion of the algorithm:

Proposition 3.4. *Given a d-regular bipartite graph G, partial matching M that leaves $2k$ vertices unmatched, and matching graph H constructed from M and G, the **expected number of steps** before a random walk from s arrives at t is at most $2 + \frac{n}{k}$.*

Proof. First, construct $H*$, a graph identical to H with the exception that vertices t and s are condensed into a single vertex $s*$. Note that now, the in- and out- degree of each vertex of $H*$ are equivalent.

Next, construct a random walk starting at $s*$; we wish to know the expected return time to $s*$ in the Markov chain defined by $H*$ (noting that $H*$ is positive recurrent because it has balanced in- and out- degrees, and is directed). From a basic course in Stochastic Processes: Expected return time in a positive recurrent Markov Chain = $\frac{1}{\text{the stationary measure}}$.

Therefore, since the stationary distribution at vertex i is

$$\pi_i = \frac{deg(i)}{\sum_{j \in V(H*)} deg(j)}$$

Then the expected return time to $s*$ of the random walk is

$$\frac{1}{\pi_i} = \frac{\sum_{j \in V(H*)} deg(j)}{deg(s*)} = \frac{(n-k)(d-1) + 2kd + kd}{kd} \leq \frac{n}{k} + 2$$

which completes the proof. □

Now, the algorithm and its correctness follow easily: a random walk from s to t, which then defines an augmenting path to improve the current matching, can be found in at most $2 + \frac{n}{k}$ steps on average.

To define the algorithm:

Algorithm 1 Truncated-Walk Subroutine

Require: vertex u, integer $b > 0$
  if $u = t$ **then**
    return SUCCESS
  end if
  $v \leftarrow$ the other endpoint of uniformly randomly chosen outgoing edge from u
  $b \leftarrow b - 1$
  if $b \leq 0$ **then**
    return FAIL
  end if
  return Truncated-Walk(v,b)

Algorithm 2 Matching Algorithm

Require: $G = (P, Q, E)$
  $j \leftarrow 0$
  $M_o \leftarrow \emptyset$
  while Perfect matching not yet found **do**
    $b_j \leftarrow 2(2 + \frac{n}{n-j})$
    while Truncated-Walk is unsuccessful **do**
      Run Truncated-Walk(s, $b + j$)
    end while
    $p \leftarrow$ the successful path from Truncated-Walk after loops are removed
    $M_{j+1} \leftarrow M_j \Delta P$
    $j \leftarrow j + 1$
  end while
  return M_j

4. Time Complexity of the Algorithm

Theorem 4.1. *The previously described algorithm for finding a perfect matching in G runs in time $O(n \log n)$ both in expectation, and with high probability.*

We will build up to the proof of this theorem with 3 lemmas.

Lemma 4.2. *The Truncated-Walk subroutine succeeds with probability $\geq \frac{1}{2}$.*

Proof. From an earlier lemma, we have that $\mathbb{E}$(steps until successful path found)$\leq 2 + \frac{n}{k}$.
Using Markov's Inequality:

$$\mathbb{P}(\text{number of steps needed} \geq b) \leq \frac{\mathbb{E}(\text{number of steps needed})}{b} \leq \frac{2+\frac{n}{k}}{b}$$

$$\Rightarrow \mathbb{P}(T\text{-}W \text{ succeeds}) \geq 1 - \frac{2+\frac{n}{k}}{b} = 1 - \frac{2+\frac{n}{n}}{2(2+\frac{n}{n-j})} = \frac{1}{2} \text{ when j=0}$$

□

Now, let X_j be the time taken by the j^{th} augmentation, and let Y_j be independent and exponentially distributed, with mean $\mu_j := \frac{b_j}{ln(2)}$

Lemma 4.3.

$$\mathbb{P}[X_j \geq qb_j] \leq \mathbb{P}[Y_j \geq qb_j]$$

Proof. From Lemma 4.2, $\mathbb{P}[X_j \geq qb_j] \leq 2^{-q}$.
By definition of the exponential distribution, $\mathbb{P}[Y_j \geq qb_j] = e^{\frac{-qb_j}{\mu_j}} = e^{\frac{-qb_j \ln(2)}{b_j}} = 2^{-q} \quad \forall \quad q > 1$ □

Using Lemma 4.2, we have that $\mathbb{P}[X_j \geq x] \leq \mathbb{P}[Y_j \geq x]$ for all $x > b_j$. So from here on, we can consider Y instead of X in finding the upper bound for runtime.

Proposition 4.4. *Let $Y := \sum_{0 \leq j \leq n-1} Y_j$. Then $Y \leq cn \log n$ with high probability for c large enough. Since this gives an upper bound for total time taken by all n augmentations, this would prove theorem 4.1.*

Proof. Let $\mu := \mathbb{E}[Y]$
Using Markov's Inequality, for any given t and δ both > 0,

$$\mathbb{P}[Y \geq (1+\delta)\mu] \leq \frac{\mathbb{E}[e^{tY}]}{e^{t(1+\delta)\mu}}$$

And if $t < \frac{1}{\mu}$, for any j,

$$\mathbb{E}[e^{tY_j}] = \frac{1}{\mu_j} \int_0^\infty e^{tx} e^{\frac{-x}{\mu_j}} = \frac{2}{1-t\mu_j}$$

And so, because the Y_j's are independent,

$$\Rightarrow \mathbb{E}[e^{tY}] = \prod_{j=0}^{n-1} \frac{1}{1-t\mu_j}$$

$$\Rightarrow \mathbb{P}[Y \geq (1+\delta)\mu] \leq \frac{\prod_{j=0}^{n-1} \frac{1}{1-t\mu_j}}{e^{t(1+\delta)\mu}} = \frac{e^{-t(1+\delta)\mu}}{\prod_{j=0}^{n-1}(1+t\mu_j)}$$

Now, we are almost done. Observing that $\mu_{n-1} > \mu_{n-1} > \cdots$, define $t := \frac{1}{2\mu_{n-1}}$, noting that this value still satisfies $t < \frac{1}{\mu_j}$ and $t > 0$ for all j. Then we have that

$$(1 - t\mu_j) \geq e^{-t\mu_j ln(4)}$$

because $1-x \geq \frac{1}{4}^x$ holds for $x \in [0, \frac{1}{2}]$ and $\mu_j t \leq \frac{1}{2}$.

Plugging this back into the previous equation:

$$\mathbb{P}[Y \geq (1+\delta)\mu] \leq \frac{e^{-\frac{(1+\delta)\mu}{2\mu_{n-1}}}}{\prod_{j=0}^{n-1} e^{-t\mu_j ln(4)}} = e^{\frac{-(1+\delta-ln4)\mu}{2\mu_{n-1}}}$$

Finally, we note that, for the n^{th} harmonic number $H(n)$,

$$\mu = \frac{2n}{ln2} + (\mu_{n-1} - \frac{2}{ln2})H(n) \geq \mu_{n-1}H(n) \geq \mu_{n-1}(ln(n))$$

since

$$\mu = \sum_{j=0}^{n-1}(4 + \frac{2n}{n-j} = 4n + 2nH(n)$$

And so, since we have seen that $\frac{\mu}{\mu_{n-1}} \geq H(n) \geq ln(n)$, μ is $O(n \log n)$ and

$$\mathbb{P}[Y \geq (1+\delta)\mu] \leq e^{\frac{-(1+\delta-ln4)\mu}{2\mu_{n-1}}} \leq n^{\frac{-(1+\delta-ln4)}{2}}$$

This gives us the required high-probability bound. □

5. An $\Omega(nd)$ Lower Bound for Deterministic Algorithms

Theorem 5.1. *For any positive integer d, any deterministic algorithm to find a perfect matching a d-regular bipartite graph requires $\Omega(nd)$ queries to the adjacency matrix of the graph, where the ordering of edges in the adjacency array is chosen by an adversary.*

Definition 5.2 (Canonical Bipartite Graph)**.** *A* ***canonical bipartite graph*** *is a graph $G(P \cup \{t\}, Q \cup \{s\}, E)$ such that:*

- *The vertex set $P = P_1 \cup P_2$ and $Q = Q_1 \cup Q_2$ where $|P_i| = |Q_i| = 2d$ for $i = 1, 2$*
- *Vertex s is connected to an arbitrary set of d distinct vertices in P_1, and vertex t is connected to an arbitrary set of d distinct vertices in Q_2*
- *G contains a perfect matching M' of size d connecting a subset $Q_1' \subset Q_1$ to a subset $P_2' \subset P_2$ where $|Q_1'| = |P_2'| = d$*
- *The rest of the edges in E connect vertices in P_i to Q_i for $i = 1, 2$ to make the degree of each vertex in G exactly d*

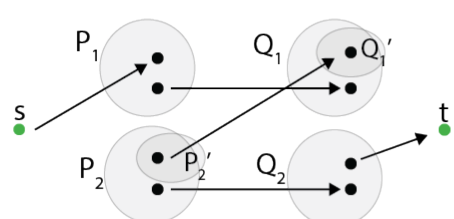

Figure 2. An example Canonical Bipartite Graph

Proof. To prove this, we will construct a family $G(d)$ of d-regular bipartite graphs with $O(d)$ vertices each, such that each graph in $G(d)$ is a canonical graph. We then want to show that there is an $\Omega(d^2)$ lower bound on queries to graphs drawn from $G(d)$. This will prove the claim, since we can take $\Theta(\frac{n}{d})$ disjoint copies of canonical graphs to create a d-regular graph on n vertices.

Let A be a deterministic algorithm for finding a perfect matching in graphs in $G(d)$. We will analyze a "game" between A and an "adaptive adversary" $\mathcal{A}$. The adversary tries to maximize the number of edges A examines before the perfect matching is found, while A tries to find an edge in M' by submitting queries to $\mathcal{A}$ about edge locations in the following manner:

- Beginning with a canonical graph G, $\mathcal{A}$ reveals s, t, the partition of vertices into P_i and Q_i (for $i = 1, 2$), and all edges (s, p) for $p \in P_1$ and (q, t) for $q \in Q_2$ at no cost to A.
- Whenever A probes a new location in the adjacency array of$u \in (P \cup Q)$, this is counted as a query $\mathcal{Q}(u)$ to $\mathcal{A}$.
- $\mathcal{A}$ responds by giving A a vertext v that has not yet been revealed as adjacent to u

Definition 5.3 (Free vertex)**.** *A vertex of graph G is **free** if its degree is $< d$.*

Lemma 5.4. *Let $G_r(P \cup \{t\}, Q \cup \{s\}, E_r)$ be a bipartite graph such that*

(1) s is connected to d distinct vertices in P_1 and t is connected to d distinct vertices in Q_2
(2) All other edges in G_r connect vertices in P_i to vertices in Q_i for some $i \in (1, 2)$. (Note that together, these comprise all the edges of a canonical graph except those in M')
(3) The degree of every vertex is $\leq d$
(4) $\exists$ at least $d + 1$ free vertices each in both Q_1 and P_2
(5) $\exists$ $u \in P_i$ and $v \in Q_i$ for some $i \in (1, 2)$ such that $(u, v) \notin G_r$

The $\exists$ a canonical graph $G(P \cup \{t\}, Q \cup \{s\}, E) \in G(d)$ such that $E_r \cup (u, v) \subset E$ iff u, v have degree $< d$ in G_r.

Proof. If either u or v have degree d, adding (u, v) violates regularity.

If u, v have degree $< d$, define G' to be G_r plus the edge (u, v). The degree of every vertex in G' is still $\leq d$. Now, to see how G' can be extended to a d-regular graph:

Add the perfect matching M' of size d to G' connecting d free vertices in Q_1 to d free vertices in P_2 (we know these vertices exist since there were at least $d + 1$ free originally, and that number decreased by at most one in adding a single extra edge.

Now, the total degree in P_i = the total degree in Q_i (for $i \in (1, 2)$). So, it is possible to pair up vertices between P_i and Q_i until every vertex has degree d. This is now a canonical graph $\in G(d)$. So, $E_r \cup (u, v) \subset E$ where E is the edge set of a canonical graph $\in G(d)$. $\square$

Finally, we can use this fact to define a strategy for the adversary.

5.1. **Adversary Strategy:** For each vertex u, $\mathcal{A}$ keeps a list $N(u)$ of all vertices that are both adjacent to u and already revealed to A.

Note that we can assume A never submits a query about a vertex for which $|N(u)| = d$, since in this case it already knows all of u's neighbors – so A only ever submits queries regarding free vertices.

At any step, define G_r to be the graph revealed to A so far.

Definition 5.5 (Evasive mode)**.** *We will say that the adversary is in **evasive mode** if G_r satisfies (1) through (4) of lemma 5.4; that is, if no edges of matching M' have been revealed to A yet.*

Observe that the game starts in evasive mode, and at some point $\mathcal{A}$ is forced to switch to non-evasive mode for the remainder of the game. Thus we define the strategy that while evasive mode is still possible, $\mathcal{A}$ responds to query $\mathcal{Q}(u)$ for $u \in P_i$ by returning a free vertex $v \in Q_i$ such that $v \notin N(u)$ (or vice versa if $u \in Q_i$). $\mathcal{A}$ then updates $N(u)$ and $N(v)$.

By Lemma 5.4, when the game becomes non-evasive, there exists a canonical graph $G \in G(d)$ that contains the graph G_r revealed so far as a subgraph. After this point occurs, the adversary answers queries arbitrarily.

Lemma 5.6. *A makes $\geq d^2$ queries before nonevasive mode begins*

Proof. In evasive mode, $\mathcal{A}$ always answers in such a way that (1) through (3) of lemma 5.4 are met. However, eventually (4) will fail when the free vertices run out. Each query contributes 1 to the degree of 1 vertex in Q_1 or P_2. Free vertices have degree $< d$ and Q_1 and P_2 start with $\geq d+1$ free vertices each, so to deplete the free vertices takes $\geq d^2$ queries.

Note that A cannot name a matching until after evasive mode ends, so the time for A to find a matching in this scheme is $\geq \Omega(d^2)$ as desired. □

This completes the proof of the proposition, and so we have proven theorem 4.1. □

References

[Bol98] Béla Bollobás. *Modern graph theory*, volume 184. Springer Science & Business Media, 1998.

[GKK09] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. Perfect matchings in o(n log n) time in regular bipartite graphs. *CoRR*, abs/0909.3346, 2009.

[GKK10] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. Perfect matchings via uniform sampling in regular bipartite graphs. *ACM Transactions on Algorithms (TALG)*, 6(2):27, 2010.

[HK73] John E Hopcroft and Richard M Karp. An nˆ5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.