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

Lecture 4: Primal Dual Matching Algorithm and Non-Bipartite Matching

Lecturer: Mohammad R. Salavatipour **Scriber:** Idanis Diaz
Date: Sept 15 and 17, 2009

We start by recalling the formulation of the maximum weight perfect matching in bipartite graphs as an integer program and the minimum weight vertex cover problem as the dual. We will see a Primal/Dual Matching Algorithm. Then we start the topic of matching in general graphs.

1 Primal/Dual Algorithm for weighted matchings in Bipartite Graphs

Recall the problem of maximum weight perfect matching in a given bipartite graph graph $G(A \cup B, E)$ with edge weights $w_{i,j} \geq 0$. The objective is to find out a maximum weight perfect matching. If we define an indicator variable $x_{i,j}$ for each edge between $i \in A$ and $j \in B$ then the integer program formulation of the weighted matching problem is:

then for $x_{i,j}$ to represent the maximum matching, we want $\sum_{i,j} x_{i,j} w_{i,j}$ to be maximized. Therefore maximum weighted matching has the following Integer Program formulation:

$$\begin{aligned} \max \quad & \sum_{i,j} x_{i,j} w_{i,j} & & & (1) \\ s.t. \quad \forall \quad a_i, \quad & \sum_j x_{i,j} & \leq & \ 1 \\ \forall \quad b_j, \quad & \sum_i x_{i,j} & \leq & \ 1 \\ & x_{i,j} & \in & \ \{0,1\} \end{aligned}$$

By relaxing the integrality constraint of IP to $x_{i,j} \geq 0$ we have a linear program. A weighted vertex cover for a graph with weighted edges is a function $y : V \to \mathbb{R}^+$ such that for all edges $e = uv$: $y_u + y_v \geq w_{uv}$. Note that for any weighted matching M and vertex cover Y:

$$\sum_{e \in M} w(e) = w(M) \leq C(Y) = \sum_{y \in V} y_v \tag{2}$$

where $C(Y)$ is the cost of the vertex cover Y. The vertex cover is actually the dual problem of maximum matching. If one considers the dual program to the LP relaxation of maximum matching problem we obtain the formulation of LP relaxation of the weighted vertex cover problem:

$$\begin{aligned} \min \quad & \sum_i y_i & & & (3) \\ s.t. \quad \forall e = a_i b_j : \quad & y_{a_i} + y_{b_j} & \geq & \ w_{i,j} \\ & y_i & \geq & \ 0 \end{aligned}$$

According to equation (2), for any vertex cover Y, $C(Y)$ is an upper bound for $w(M)$ for any matching M. So if a vertex cover is founded and has the same value as a matching both are optimal solutions, i.e.

Lemma 1.1 *For a perfect matching M and a weighted vertex cover y:*

$$C(y) \geq W(M)$$

Also $C(y) = W(M)$ iff *M consists of edges $a_i b_j$ such that $y_i + y_j = w_{i,j}$. In this case M is optimum.*

1.1 The Primal/Dual Algorithm

The algorithm starts with $M = \emptyset$ and a trivial feasible solution for the weighted vertex cover which is the following one:

$$\forall \quad a_i \in A; \quad y_{a_i} = \max_j \quad w(a_i b_j)$$
$$\forall \quad b_j \in B; \quad y_{b_j} = 0$$

At any iteration of the algorithm we build an equality graph defined below.

The *equality graph*, $G_y = (A \cup B, E_y)$ is built based on the y values such that it only contains *tight* edges

$$a_i b_j \in E_y \quad \Leftrightarrow \quad y_i + y_j = w_{i,j}$$

Let us call $y_{a_i} + y_{b_j} - w_{i,j}$, the *excess* of $a_i b_j$.

Observation 1.2 *If M is a perfect matching in G_y then $W(M) = \sum_{a_i \in A} y_{a_i} + \sum_{b_j \in B} y_{b_j}$ and by previous lemma that matching in G is an optimum solution.*

Based on this observation, the goal of the algorithm is to find a perfect matching in the equality graph. For this we update y to make more edges tight to be added to E_y (until it contains a perfect matching) while keeping y a vertex cover. Now we present an algorithm to add an edge to equality graph.

Suppose we are at some iteration of the algorithm and M is a maximum matching in G_y but is not perfect. Construct digraph D as before[1]. Let L be a set of nodes accessible from any exposed node in A. Recall that $C^* = (A - L) \cup (B \cap L)$ is a vertex cover. Therefore there is no edge between $A \cap L$ and $B - L$ (otherwise that edge is not covered by C^*). However, we know that we start with a complete graph G. Thus there are edges in G between $A \cap L$ and $B - L$ but they are not in G_y; which means all those edges have positive excess (i.e. are not tight). We update the y values to make one of these edges go tight. Let

$$\epsilon = \min\{y_{a_i} + y_{b_j} - w_{i,j} \quad s.t., \quad a_i \in A \cap L, \quad b_j \in B - L\}$$

be the minimum excess value of all such edges (these are the edges that could be added to G_y). Then we update vertex y values to tighten the edges with ϵ excess value by defining:

$$y_{a_i} = \begin{cases} y_{a_i} & a_i \in A - L \\ y_{a_i} - \epsilon & a_i \in A \cap L \end{cases}$$

and

$$y_{b_j} = \begin{cases} y_{b_j} & b_j \in B - L \\ y_{b_j} + \epsilon & b_j \in B \cap L \end{cases}$$

Note that by this change every edge that was in G_y remains tight. Also by the choice of ϵ, no edge constraint is going to be violated, so y remains a vertex cover. Furthermore, at least one edge between $A \cap L$ and

[1] $D = (A \cup B, E')$, where E' is the union of edges of M directed from B to A and edges of $E - M$ directed from A to B.

Maximum Weighted Bipartite Matching Algorithm; Primal-Dual Method

For each $a_i \in A$ let $y_{a_i} = \max_{b_j \in B} w(a_i b_j)$;
For each $b_j \in B$ let $y_{b_j} = 0$;
Build graph G_y and let M be a maximum matching in G_y
Construct Digraph D
repeat
    let L be the set of nodes (in D) accessible from any exposed node in A.
    Let $\epsilon = \min\{y_{a_i} + y_{b_j} - w(ij) : \quad a_i \in A \cap L, \quad b_j \in B - L\}$
    Decrease y_{a_i} for each $a_i \in A \cap L$ by ϵ and increase y_{b_j} for each $b_j \in B - L$ by ϵ
    Add the tight edges to G_y and recompute matching M.
Until M is a perfect matching.

$B - L$ goes tight and therefore is added to G_y. We can repeat this operation until either G_y has a perfect matching, or there is no edges left between $A \cap L$ and $B - L$. The latter happens only if both these sets are empty, which implies that we have a perfect matching. Thus, eventually we find a perfect matching in G_y which by the previous lemma corresponds to an optimum matching in G. At this point the solution y is also an optimum vertex cover. The following pseudo code summarize the Primal/Dual Algorithm:

At each iteration of the algorithm at least one edge is added to G_y, so there are $O(n^2)$ iterations. To update the maximum matching in G_y and re-compute L we have to spend at most $O(n^2)$ time; thus the total running time is bounded by $O(n^4)$. In fact a more careful analysis shows that the running time can be bounded by $O(n^3)$. Thus:

Theorem 1.3 *Given a complete bipartite graph G with edge weights $w(e)$, the primal/dual method finds a maximum matching and a minimum vertex cover in time $O(n^3)$.*

2 Non-Bipartite Matching

So far we have been talking about matchings (and weighted matchings) in bipartite graphs. Finding matchings in general (non-bipartite) graphs is substantially more difficult.

In bipartite graphs, König's theorem is a min-max theorem between the maximum size of a matching and the minimum size of a vertex cover. Although vertex cover is the dual problem, this min-max relation does not hold in general graphs (a simple example being a cycle of length 3). However we can still give min-max theorems for general graphs.

Definition 2.1 *Given a set of vertices $U \subseteq V$, $G - U$ is the subgraph obtained by deleting all the nodes of U and their incident edges.*

Definition 2.2 *An odd component is a connected component with odd number of nodes. We use $o(G)$ to denote the number of odd components of graph G.*

Suppose M is a matching in $G - U$ and consider an odd component of $G - U$. There is at least one un-matched node in this component. Therefore, to have a perfect matching in G, each such vertex (in an odd component) must be matched to a node in U. But we can add at most U matching edges of this type. Therefore we must have $o(G - U) \leq |U|$ in order to have a perfect matching.

Tutte proved that this necessary condition is also sufficient.

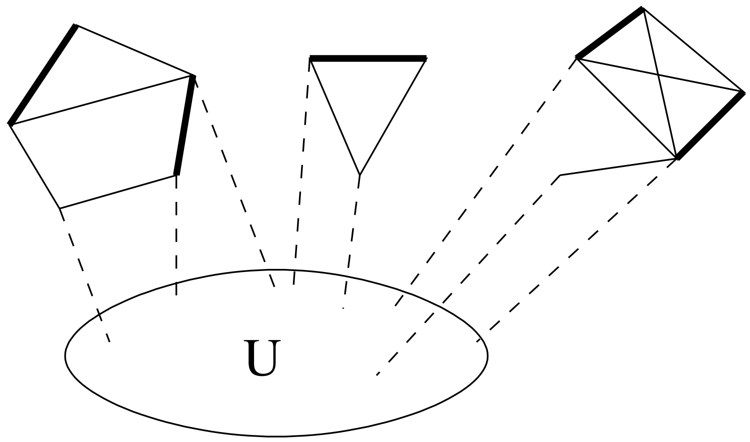


Figure 1: a matching in $G-U$ and the odd components of $G-U$

Theorem 2.3 (Tutte) *A graph G has a perfect matching iff* $o(G-U) < |U| \quad \forall\, U \subseteq V$.

What if G does not have a perfect matching? It is easy to see that each matching in G has at least $o(G-U) - |U|$ unmatched edges. This shows that the size of a maximum matching is at most $(|V|+|U|-o(G-U))/2$. It turns out that the minimum value of this (over all sets U) is actually equal to the size of a maximum matching. If we use $\nu(G)$ to denote the size of a maximum matching of G then:

Theorem 2.4 (Tutte-Berge Formula)

$$\nu(G) \max_{M} |M| = \min_{U \subseteq V} \frac{|V| - o(G-U) + |U|}{2} \tag{4}$$

Proof: Assume that G is connected (otherwise we can prove it for each connected component of G). It is clear that the r.h.s. is an upper bound for the size of any matching. We prove the other direction by induction on $|V|$. The cases $|V| \leq 1$ is trivial. So we assume $|V| \geq 24$. Two cases are considered:

Case 1: If G has a vertex $v \in V$ saturated by all maximum matchings of G. Then the size $\nu(G-v) = \nu(|G|) - 1$ and by induction there is a set $U' \subseteq V - v$ such that:

$$\nu(G-v) = \frac{1}{2}(|V-v| + |U'| - o(G-v-U').$$

Define $U = U' \bigcup \{v\}$. Then:

$$\begin{aligned}
\nu(G) &= \nu(G-v) + 1 \\
&= \frac{1}{2}(|V-v| + |U'| - o(G-v-U')) + 1 \\
&= \frac{1}{2}(|V| - 1 + |U| - 1 - o(G-U)) + 1 \\
&= \frac{1}{2}(|V| + |U| - o(G-U)).
\end{aligned}$$

Case 2: So suppose that there is no such vertex v. The for each $v \in V$ there is some maximum matching not covering v. Therefore $\nu(G) < \frac{1}{2}|V|$. We show that there is exactly one vertex missed in any maximum matching and therefore there is a matching of size $\frac{1}{2}(|V|-1)$ which implies the theorem with $U = \emptyset$. By way of contradiction, suppose there are two distinct vertices not covered by a maximum matching M, call them u, v. Among all such triples M, u, v choose one such that $d(u, v)$ is the smallest. If $d(u, v) = 1$, then u and v are adjacent, so we can augment M by the edge uv, contradicting the maximality of M; so $d(u, v) \geq 2$. Therefore, there is an intermediate vertex t on the path between u and v. Because we are in case 2, there is a maximum matching N missing t and $N \neq M$ (otherwise M misses both u and t which contradicts definition of M, u, v). Among all such maximum matchings N missing t choose one such that $|M \bigcap N|$ is the largest. Observe that N must cover u and v. Since both N and M are maximum matchings the number of vertices covered by both is the same. Consequently, there is a vertex $x \neq t$ covered by M but not by N. Let $e = xy \in M$; y must be saturated by N (otherwise e could be added to N). In other words y is covered by some edge $f \in N$, $e \in M$, and $e \notin N$. Now we can modify N to $N' = N - f + e$ in this case the intersection of N' with M increases is larger than that of N, contradicting the choice of N. ■

One might think that the idea of using augmenting paths to find maximum matchings could be extended to general graphs. This is true as shown below:

Theorem 2.5 (Berge) *A matching M is maximum iff there is not M-augmenting path.*

Proof: Clearly if there is an M-augmenting path it is not a maximum matching. We prove the other direction. Suppose that M is not maximum and there is not any augmenting path. Let M^* be a maximum matching with the largest $|M \bigcap M^*|$. Consider the union $M \bigcup M^*$. Observe that this union cannot have even paths or cycles (as otherwise we could reverse the edges of them to make $M^* \cap M$ larger). Therefore, there can only be odd paths with more edges from M^* then M. Such a path is clearly is an M-augmenting path. ■

The major question is how to find an M-augmenting path. A natural approach could be to try to use the same idea as in bipartite graphs to find an augmenting path. The difficulty is that since the graph is not bipartite we don't know how to orient the edges to find the augmenting path. The presence of odd cycles in a general graphs makes it particularly difficult as an alternating path can cross itself. It was Jack Edmonds' who came up with a clever idea to handle this difficulty.

Definition 2.6 *Let M be a matching in G, and X be the set of vertices missed by M. An M-alternating walk $P = (v_0, v_1, ..., v_t)$ is called an M-flower if t is odd and $(v_0, v_1, ..., v_{t-1})$ are distinct, $v_0 \in X$, and $v_t = v_i$ for some even $i < t$. With this condition the set of vertices $v_i, ..., v_t$ are called blossom and the vertices $v_0, .., v_i$ are called stem.*

Figure (2) shows a flower. The vertices $v_0, \ldots, v_4$ represent the stem part while the vertices $v_4, \ldots, v_8$ represent the blossom part. We can easily reverse the edges of the stem so that $v_i = v_t \in X$ without changing the size of the matching; this way we obtain an M-flower which consists of just a blossom (i.e. the stem is empty).

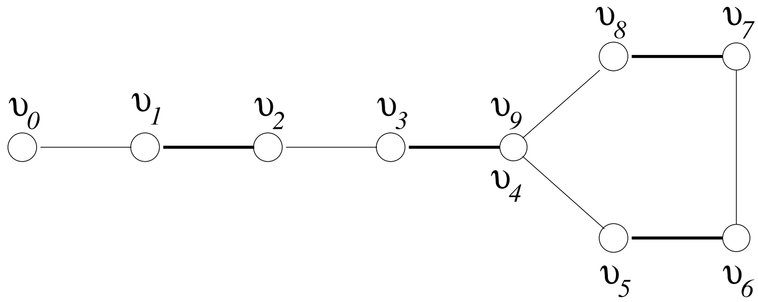
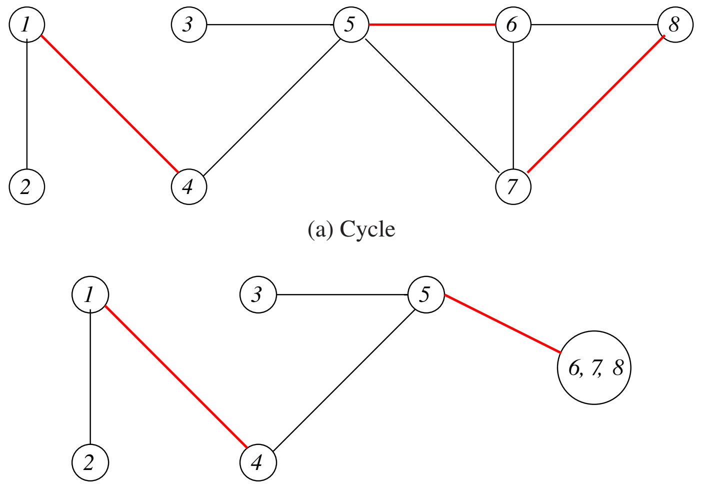


Figure 2: A graph containing a stem and blossom

The core of Edmonds' algorithm is based on the following observation. Let $B \subseteq V$ be a subset of vertices.

Then G/B is the graph G' obtained by shrinking the set B into a single node b; G' consists of $(V - B) \bigcup \{b\}$ and each edge $e \in G'$ is obtained by replacing an end-vertex in B with the new node b. We call the new edges of G' the images of the original edges in G. As an example, consider the graph in Figure (3a). In the figure (3a) the vertices $6, 7, 8$ constitute a cycle while the figure (3b) illustrates how the vertices $6, 7, 8$ were shrunk in a single node which correspond to a new node b.



(a) Cycle

(b) Shrunk nodes

Figure 3: Graphs containing the illustration of passing from a cycle to shrunk node

For any matching M, let M/B denote the set of edges in G/B that are images of edges in M not spanned by B (i.e. have at least one end-point not in B). If M intersects $\delta(B)$ in at most one edge then M/B is a matching in G/B. So if we can find a blossom in G and shrink it into a single node and find a maximum matching in graph G/B then we can extend it to a maximum matching in G. We formally prove this.

Theorem 2.7 *Let B be an M-Blossom in G. Then M is a maximum matching in G iff M/B is a maximum matching in G/B.*

Proof: Let $B = (v_0, \ldots, v_t)$ (we assume this is a blossom of a flower with empty stem). First assume that M/B is not a maximum size matching in G/B. So there is an M/B-augmenting path P in G/B. If P does not involve vertex b of G/B then it is an M-augmenting path in G and so M is maximum. So suppose P enters b using some edge ub not in the matching M/B. then $uv_j \in E$ for some $j \in \{0, 1, .., t\}$.

- If j is even then replace b in P with $v_j, v_{j-1}, \ldots, v_0$
- if j is odd then replace b in P with $v_j, v_{j+1}, ..., v_t$

In both cases an augmenting path is obtained in G.

Now suppose that M is not maximum, but M/B is maximum. There is an M-augmenting path $P = u_0, u_1, \ldots, u_s$. If P does not involve B then it is an augmenting path in G/B and we are done. So say P it intersects B and let u_j be the first vertex of P in B. Then $u_0, u_1, \ldots, u_j$ is an M/B-augmenting path in G/B; thus M/B is not maximum. ■

The following theorem combined with the previous two will complete the tools needed for our matching algorithm.

Theorem 2.8 *Let M be a matching in G and $P = v_0, v_1, ...v_t$ be a shortest M-alternating walk from a node in X to another node in X (i.e. a shortest $X - X$ walk). Then either P is an M-augmenting path or*

$v_0, ..., v_t$ *is an* M*-flower for some* $j < t$

Proof: Assume that P is not a path. So there are $i < j$ with $v_j = v_i$; let j be the smallest such index. Therefore, $v_0, ..., vj_1$ are all distinct. We prove that $v_0, \ldots, v_j$ is an M-flower. If $j - i$ is even then we can delete $v_{i+1}, ..., v_j$ from P to obtain a shorter M-alternating $X - X$ walk. So $j - i$ is odd. If j is even and i is odd, then $v_{i+1} = v_{j-1}$ (it is a the vertex matched to $v_i = v_j$), contradicting the minimality of j. Thus j is odd and i is even and therefore $(v_0, v_1, ..v_j)$ is an M-flower. ∎

Using these theorems we can describe Edmonds' blossom algorithm for general matching. We use the following procedure to iteratively find augmenting paths as long as there is one:

The Matching Augmenting Algorithm

$X \leftarrow$ Exposed nodes
If there is no alternating $X - X$ walk then
    return No augmenting path
Else
    Let P be a shortest $X - X$ alternating walk
    If P is a path return P
    Else
        It contains a blossom B
        Compute G/B and find an M/B-augmenting path P' in G/B
        Expand P' to an augmenting path in G
        Return P

Theorem 2.9 *Given a graph G, a maximum matching can be found in* $O(n^2m)$.

Proof: Theorems 2.5, 2.7, 2.8 prove the correctness of the above algorithm. An M-alternating walk can be found in $O(m)$ time using a BFS. This either yields an M-augmenting path or a blossom B. In the latter case the shrunk graph G/B can be built in $O(m)$ as well. The number of recursive calls is at most $O(n)$ since each time the size of the graph decreases by a constant. Thus in $O(mn)$ we can find an M-augmenting path if one exists. There are a total of $O(n)$ iterations of calling this procedure and so the total running time is $O(n^2m)$. This can be improved to $O(n^3)$ as well. ∎

References

1. Combinatorial Optimization, Schrijver, (Volume 1) Springer-Verlag, 2003.