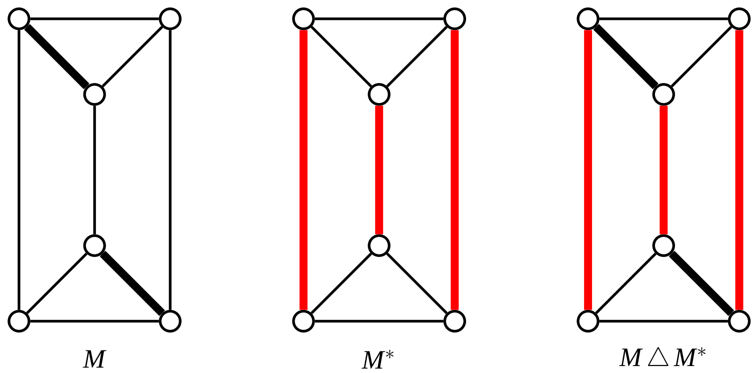


Figure 11: Illustration of the existence of an M-augmenting path.

7. Matchings

7.1 Introduction

Recall that a *matching* M in an undirected graph $G = (V,E)$ is a subset of edges satisfying that no two edges share a common endpoint. More formally, $M \subseteq E$ is a matching if for every two distinct edges $(u,v),(x,y) \in M$ we have $\{u,v\} \cap \{x,y\} = \emptyset$. Every node $u \in V$ that is incident to a matching edge is said to be *matched*; all other nodes are said to be *free*. A matching M is *perfect* if every node $u \in V$ is matched by M.

We consider the following optimization problem:

Maximum Matching Problem:

Given:	An undirected graph $G = (V,E)$.
Goal:	Compute a matching $M \subseteq E$ of G of maximum size.

Note that if the underlying graph is bipartite, then we can solve the maximum matching problem by a maximum flow computation.

Given two sets $S,T \subseteq E$, let $S \triangle T$ denote the *symmetric difference* of S and T, i.e., $S \triangle T = (S \setminus T) \cup (T \setminus S)$.

7.2 Augmenting Paths

Given a matching M, a path P is called *M-alternating* (or simply *alternating*) if the edges of P are alternately in M and not in M. If the first and last node of an M-alternating path P are free, then P is called an *M-augmenting* (or *augmenting*) path. Note that an augmenting path must have an odd number of edges. An M-augmenting path P can be used to increase the size of M: Simply make every non-matching edge on P a matching edge and vice versa. We also say that we *augment M along P*.

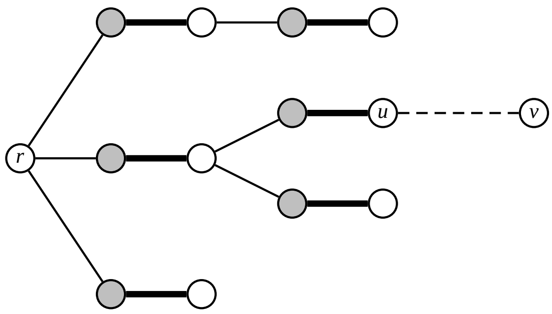


Figure 12: Illustration of an alternating tree. The nodes in X and Y are indicated in white and gray, respectively. Note that there is an augmenting path from r to v.

Theorem 7.1. *A matching M in a graph $G = (V,E)$ is maximum if and only if there is no M-augmenting path.*

Proof. Suppose M is maximum and there is an M-augmenting path P. Then augmenting M along P gives a new matching $M' = M \triangle P$ of size $|M|+1$, which is a contradiction.

Suppose that M is not maximum. Let M^* be a maximum matching. Consider the symmetric difference $M \triangle M^*$. Because M and M^* are matchings, the subgraph $G' = (V, M \triangle M^*)$ consists of isolated nodes and node-disjoint paths and cycles. The edges of every such path or cycle belong alternately to M and M^*. Each cycle therefore has an even number of edges. Because $|M^*| > |M|$ there must exist one path P that has more edges of M^* than of M. P is an M-augmenting path; see Figure 11 for an illustration. $\square$

7.3 Bipartite Graphs

The above theorem gives an idea how to compute a maximum matching: Start with the empty matching $M = \emptyset$. Find an M-augmenting path P and augment M along P. Repeat this procedure until no M-augmenting path exists and M is maximum.

A natural approach to search for augmenting paths is to iteratively build an *alternating tree*. Suppose M is a matching and r is a free node. We inductively construct a tree T rooted at r as follows. We partition the node set of T into two sets X and Y: For every node $u \in X$, there is an even-length alternating path from r to u in T; for every node $u \in Y$, there is an odd-length alternating path from r to u in T. We start with $X = \{r\}$ and $Y = \emptyset$ and then iteratively extend T using the following operation:

EXTEND TREE USING (u,v):
(Precondition: $(u,v) \in E$, $u \in X$, $v \notin X \cup Y$ and $(v,w) \in M$)
Add edge (u,v) to T, v to Y, edge (v,w) to T and w to X

This way we obtain a layered tree rooted at r (starting with layer 0); see Figure 12 for an illustration. All nodes in X are on even layers and all nodes in Y are on odd layers. Moreover, every node in layer $2i-1$ ($i \geq 1$) is matched to a node in layer $2i$. In particular, $|X| = |Y| + 1$.

Input: undirected bipartite graph $G = (V, E)$.
Output: maximum matching M.

Initialize: $M = \emptyset$
foreach $r \in V$ **do**
    if *r is matched* **then** continue
    else
        $X = \{r\}$, $Y = \emptyset$, $T = \emptyset$
        while *there exists an edge* $(u, v) \in E$ *with* $u \in X$ *and* $v \notin X \cup Y$ **do**
            if *v is free* **then** AUGMENT MATCHING USING (u, v)
            else EXTEND TREE USING (u, v)
        end
    end
end
return M

Algorithm 12: Augmenting path algorithm.

Suppose that during the extension of the alternating tree T we encounter an edge $(u, v) \in E$ with $u \in X$ and $v \notin X \cup Y$ being a free node. We have then found an augmenting path from r to v; see Figure 12.

AUGMENT MATCHING USING (u, v)
(Precondition: $(u, v) \in E$, $u \in X$, $v \notin X \cup Y$ free)
Augment M along the concatenation of the r, u-path in T with edge (u, v)

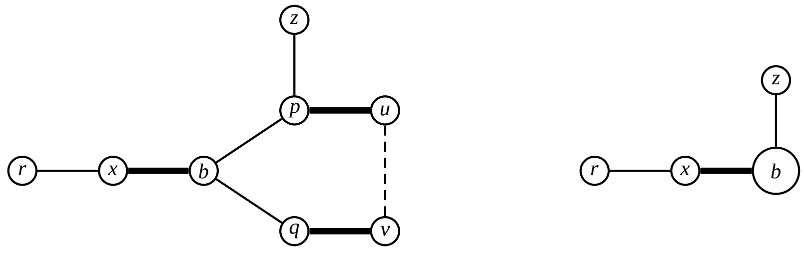
These two operations form the basis of the augmeting path algorithm given in Algorithm 12.

The correctness of the algorithm depends on whether alternating trees truly capture all augmenting paths. Clearly, whenever the algorithm finds an augmenting path starting at r, this is an augmenting path. But can we conclude that there is no augmenting path if the algorithm does not find one? As it turns out, the algorithm works correctly if the underlying graph satisfies the *unique label property*: A graph satisfies the *unique label property* with respect to a given matching M and a root node r if the above tree building procedure uniquely assigns every node $u \in V(T)$ to one of the sets X and Y, irrespective of the order in which the nodes are examined.

Lemma 7.1. *Suppose a graph satisfies the unique label property. If there exists an M-augmenting path, then the augmenting path algorithm finds it.*

Proof. Let $P = \langle r, \ldots, u, v \rangle$ be an augmenting path with respect to M. Because of the unique label property, the algorithm always ends up with adding node u to X and thus discovers an augmenting path via edge (u, v). $\square$

Using the above characterization, we can show that the augmenting path algorithm given in Algorithm 12 is correct for bipartite graphs: Recall that in a bipartite graph, the node set V is partitioned into two sets V_0 and V_1. Every node that is part of $V(T)$ and belongs



(a) (b)

Figure 13: Illustration of a blossom shrinking. (a) The odd cycle $B = \langle b, p, u, v, q, b \rangle$ constitutes a blossom with base b and stem $\langle r, x, b \rangle$. Note that there is an augmenting path from z to r via edge (u, v). (b) The resulting graph after shrinking blossom B into a super-node b.

to the set V_i with $r \in V_i$ is added to X; those that belong to V_{1-i} are added to Y. Thus bipartite graphs satisfy the unique label property.

Theorem 7.2. *The augmenting path algorithm computes a maximum matching in bipartite graphs in time $O(nm)$.*

Proof. The correctness of the algorithm follows from the discussion above. Note that each iteration can be implemented to run in time $O(n+m)$ and there are at most n iterations. □

7.4 General Graphs

It is not hard to see that graphs do in general not satisfy the unique label property. Consider an odd cycle consisting of three edges $(r, u), (u, v), (v, r)$ and suppose that $(u, v) \in M$ and r is free. Then the algorithm adds u to Y if it considers edge (r, u) first, while it adds u to X if it considers edge (r, v) first. Odd cycles are precisely the objects that cause this dilemma (and which are not present in bipartite graphs).

A deep insight that was first gained by Edmonds in 1965 is that one can "shrink" such odd cycles. Suppose during the construction of the alternating tree, the algorithm encounters an edge (u, v) with $u, v \in X$; see Figure 13(a). Let b be the lowest common ancestor of u and v in T. Note that $b \in X$. Consider the cycle B that follows the unique b, u-path in T, then edge (u, v) and then the unique v, b-path in T. B is an odd length cycle, which is also called a *blossom*. The node b is called the *base* of B. The even length path from b to the root node r is called the *stem* of B; if $r = b$ then we say that the stem of B is empty. Suppose we shrink the cycle B to a super-node, which we identify with b; see Figure 13(b). Note that the super-node b belongs to X after shrinking.

SHRINK BLOSSOM USING (u, v):

(Precondition: $(u, v) \in E$ and $u, v \in X$)

Let b be the lowest common ancestor of u and v in T.

Shrink the blossom $B = \langle b, \ldots, u, v, \ldots, b \rangle$ to a super-node b.