Lecture 2: augmenting path algorithm

Dabeen Lee

Industrial and Systems Engineering, KAIST

2025 Winter Lecture Series on Combinatorial Optimization

January 13, 2025

Outline

- Augmenting path algorithm
- Alternating tree procedure

Goal of this lecture

- The greedy algorithm that finds a maximal matching in a bipartite graph whose size is always at least half of the maximum size of a matching.
- Hence, the greedy algorithm is a constant **approximation algorithm**.
- In this section, we will introduce an algorithm that is guaranteed to compute a maximum matching of a bipartite graph.
- The central idea of the algorithm lies in the concept of **augmenting paths**, so the algorithm is referred to as the **augmenting path algorithm**.

Augmenting paths

- Let $G = (V, E)$ be a bipartite graph, and let M be a matching of G.
- We say that a vertex $v \in V$ is M-**exposed** if v is not connected to an edge in M.
- We say that a sequence of edges $e_1, \ldots, e_k$ is M-**alternating** if for every two consecutive edges e_i and e_{i+1}, either $e_i \in M, e_{i+1} \notin M$ or $e_i \notin M, e_{i+1} \in M$ holds.

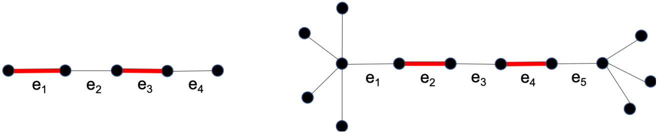


Figure: an M-alternating path and an M-augmenting path

- An M-**augmenting path** is an M-alternating path if the first and last vertices are M-exposed.

Augmenting paths

- The key idea is that if there is an M-augmenting path, we can improve the matching.

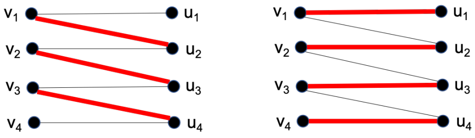


Figure: improving the matching by an augmenting path

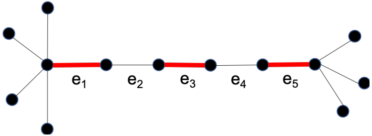
- On the augmenting path, we switch the role of the matching edges and that of the edges not in the matching.
- In other words, we remove every edge $e \in M$ from M and add every edge $e \notin M$ to M.

Augmenting paths

- To formalize the idea, we take an M-augmenting path P. We define the **symmetric difference** of M and P, denoted $M \oplus P$, as

$$M \oplus P = (M \setminus P) \cup (P \setminus M).$$

- Hence, an edge $e \in E$ belongs to $M \oplus P$ if and only if e is contained in precisely one of M and P.
- Taking the symmetric difference of the matching M and an M-augmenting path P, a change is made only on the edges of P.
- If $P = e_1, \ldots, e_{2\ell-1}$ for some $\ell \geq 1$ with $e_1, e_3 \ldots, e_{2\ell-1} \notin M$ and $e_2, e_4, \ldots, e_{2\ell-2} \in M$, we get $e_1, e_3 \ldots, e_{2\ell-1} \in M$ and $e_2, e_4, \ldots, e_{2\ell-2} \notin M$ after taking the synmetric difference.



- Then P becomes an alternating path with one more matching edge.

Augmenting paths

Lemma

Let $G = (V, E)$ be a graph, not necessarily bipartite. Let M be a matching, and let P be an M-augmenting path. Then $M \oplus P$ is a matching of G with $|M \oplus P| = |M| + 1$.

Augmenting path algorithm

- The lemma leads to a natural algorithm that iteratively improves the given matching for a bipartite graph by finding an augmenting path.

Algorithm 1 Augmenting path algorithm for maximum bipartite matching

---

Initialize $M = \emptyset$.
while there is an M-augmenting path **do**
    Find an M-augmenting path P
    Update M as $M = M \oplus P$
end while
Return M

Correctness

Theorem

Let $G = (V, E)$ be a graph, not necessarily bipartite, and let M be a matching. Then M is a maximum matching if and only if there is no M-augmenting path in G.

Correctness

Computational complexity

- In addition to its correctness, we also care about its **computational complexity**, measuring the amount of computational costs to terminate.
- For the augmenting path algorithm, we will analyze its **time complexity** or **iteration complexity**.
- Recall that each augmenting path increases the size of a matching by 1 while the maximum size of a matching is at most $|V|/2$.
- Therefore, the number times the while loop is incurred is at most $|V|/2$.
- Then, what remains is to analyze the computational complexity of finding an M-augmenting path.
- We will show that an M-augmenting path can be found in $O(|E|)$ time.

Trees

- We say that two distinct vertices u and v are **connected** if there is an uv-path.
- We say that a graph is connected if any of its two distinct vertices are connected.
- A connected graph is a **tree** if any two distinct vertices are connected by exactly one path.

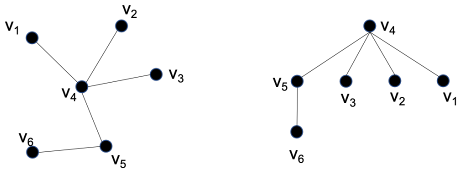


Figure: a tree and its hierarchical representation

- The terminology comes from the fact that a tree can be depicted in a hierarchical fashion.
- First, we take any vertex v as a **root** and expand the tree with its neighbors.
- We say that a vertex of degree 1 in a tree is a **leaf** vertex.
- Then we take all leaf vertices and add their neighbors to the tree on the next level.

Forests

- The **connected components** of a graph are its maximal connected subgraphs.
- A **forest** is a graph all of whose connected components are trees.

Exercise

A graph is a forest if and only if it has no cycle as a subgraph.

Alternating tree procedure

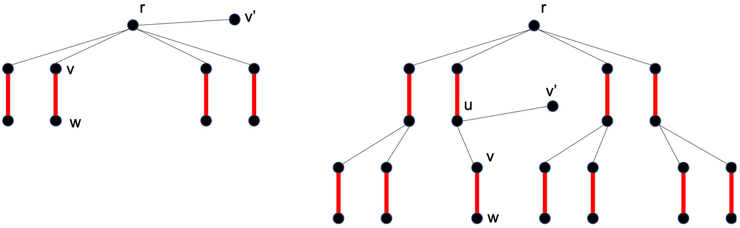
Alternating tree procedure

Algorithm 2 Alternating tree algorithm to find an M-augmenting path

Input: a bipartite graph $G = (V, E)$ and a matching M
while there is an M-exposed vertex in G **do**
  Take an M-exposed vertex r and set it as the root.
  Initialize $T = \{r\}$ and $L = \{r\}$
  while $L \neq \emptyset$ **do**
    Take a vertex $u \in L$
    if u has a neighbor that is M-exposed **then**
      Return the path from the root to u on T.
    else
      for every neighbor v **do**
        Take the vertex w such that $vw \in M$
        Update $T = T \cup \{v, w\}$ and $L = (L \setminus \{u\}) \cup \{w\}$
      end for
    end if
  end while
  Delete all vertices in T from G
end while

Alternating tree procedure

- The algorithm builds a tree structure starting from an M-exposed vertex as its root.



- We call such a tree an M-**alternating tree**.

Alternating tree procedure

Theorem

Let $G = (V, E)$ be a bipartite graph, and let M be a matching. If Algorithm 2 does not return an M-augmenting path, then G contains no M-augmenting path as a subgraph.

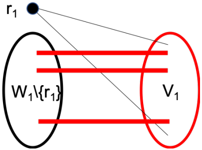


Figure: the first M-alternating tree

Alternating tree procedure

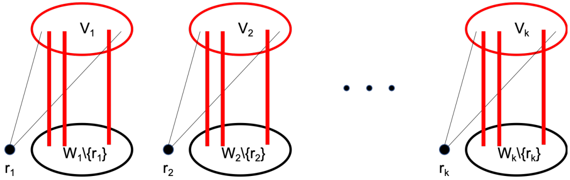


Figure: a partition of V with M-alternating trees

Alternating tree procedure

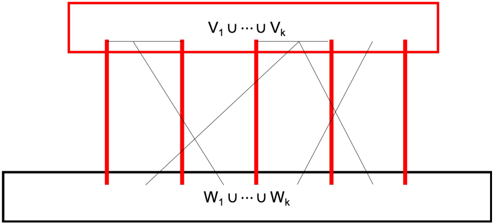


Figure: another illustration of the partition

Computational complexity

- Note that an edge is enumerated when one of its endpoints is part of an alternating tree.
- Hence, an edge is considered at most twice while running the alternating tree procedure.
- Therefore, the number of iterations required is $O(|E|)$.
- Recall that the number of while loops incurred for Algorithm 1 is $O(|V|)$.
- As a result, the computational complexity of Algorithm 1 is $O(|V||E|)$.