Lecture 3: linear programming for bipartite matching

Dabeen Lee

Industrial and Systems Engineering, KAIST

2025 Winter Lecture Series on Combinatorial Optimization

January 14, 2025

Outline

- (Recap) Augmenting path algorithm with the alternating tree procedure.
- Maximum weight bipartite matching.
- Linear programming-based method.
- (If time allows) Perfect matching and Hall's marriage theorem.

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 path with 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.

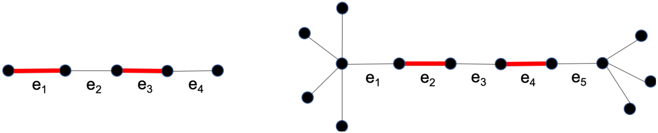


Figure: an M-alternating path and an M-augmenting path

- 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.

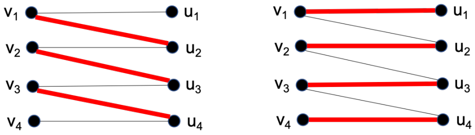


Figure: improving the matching by an augmenting path

- On the augmenting path, we switch the role of the matching edges and that of the edges not in the matching.

Augmenting paths

- Given a matching M and an M-augmenting path P, their **symmetric difference** $M \oplus P$ is obtained from M after augmenting the edges of P.

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 $|M \oplus P| = |M| + 1$.

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.

Augmenting path algorithm

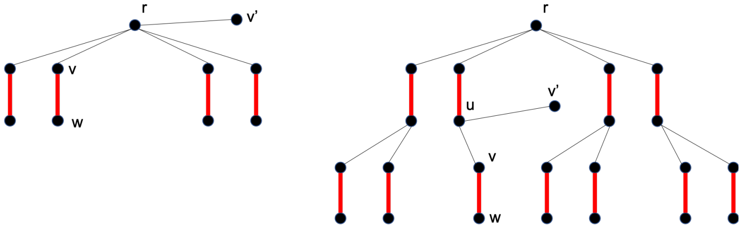
Algorithm 1 Augmenting path algorithm for maximum bipartite matching

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

- As an augmenting path increases the matching size by 1, the algorithm finds at most $|V|/2$ augmenting paths.

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 **??** *does not return an M-augmenting path, then G contains no M-augmenting path as a subgraph.*

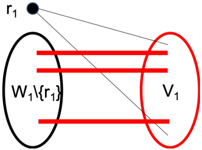


Figure: the first M-alternating tree

Alternating tree procedure

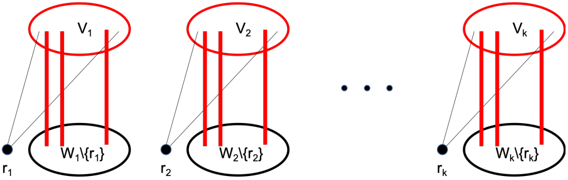


Figure: a partition of V with M-alternating trees

Alternating tree procedure

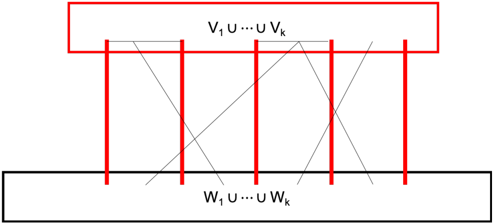


Figure: another illustration of the partition

Maximum weight matching

- Maximum matching seeks to **maximize the number of edges** in a matching, in which individual edges are treated equally.
- Some edges can be more important than others, captured by **edge weights**.
- With the edge weights, an alternate objective is to find a matching that **maximizes the total weight sum of its edges**.
- New objective:

$$\text{maximize} \quad \sum_{e \in M} \underbrace{w_e}_{\text{the weight of edge } e}.$$

- This is referred to as the **maximum weight bipartite matching** problem.
- One may extend the augmenting path algorithm to the weighted case, let us discuss another method that has a slightly different flavor.

Formulating as an optimization problem

Variables

- For each edge $e \in E$, use **variable** x_e to indicate whether e is picked for our matching M or not, i.e.,

$$x_e = \begin{cases} 1 & \text{if } e \text{ is included in matching } M, \\ 0 & \text{otherwise.} \end{cases}$$

- In this case, $x \in \{0, 1\}^{|E|}$ is the **incidence vector**, or the **characteristic vector**, of matching M given by

$$M = \{e \in E : x_e = 1\}.$$

- Then we have

$$\sum_{e \in M} w_e = \sum_{e \in E} w_e x_e.$$

Formulating as an optimization problem

Constraints

- However, not all $x \in \{0,1\}^{|E|}$ corresponds to a matching.
- What we know is that a vertex $u \in V$ is incident to at most one edge of a matching.
- We may impose the condition by setting

$$\sum_{v \in V : uv \in E} x_{uv} \leq 1 \qquad \text{(degree)}$$

where the sum is taken over the neighbors of u.

Lemma

Let $G = (V, E)$ be a graph, not necessarily bipartite, and let $x \in \{0,1\}^{|E|}$. Then x satisfies (degree) for all $u \in V$ if and only if x is the incidence vector of some matching M of G.

Formulating as an optimization problem

Optimization problem

- Then the following computes a maximum weight matching:

$$\begin{aligned} \text{maximize} \quad & \sum_{e\in E} w_e x_e \\ \text{subject to} \quad & \sum_{v\in V: uv\in E} x_{uv} \leq 1 \quad \text{for all } u \in V, \\ & x_e \in \{0,1\} \quad \text{for all } e \in E. \end{aligned} \tag{IP}$$

- This is to select a vector x that achieves the maximum value of $\sum_{e\in E} w_e x_e$ among the vectors satisfying (degree) for all $u \in V$ and $x \in \{0,1\}^{|E|}$.
- For (IP), constraint (degree) is referred to as the **degree constraint**.
- Constraint $x \in \{0,1\}^{|E|}$ is called the **binary constraint**.

Formulating as an optimization problem

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} w_e x_e \\ \text{subject to} \quad & \sum_{v \in V : uv \in E} x_{uv} \leq 1 \quad \text{for all } u \in V, \\ & x_e \in \{0,1\} \quad \text{for all } e \in E. \end{aligned} \tag{IP}$$

Proposition

Let $G = (V, E)$ be a graph, not necessarily bipartite, and let $w \in \mathbb{R}^{|E|}$. Then solving the optimization problem (IP) computes a maximum weight matching in G.

Formulating as an optimization problem

$$\begin{aligned} \text{maximize} \quad & \sum_{e\in E} w_e x_e \\ \text{subject to} \quad & \sum_{v\in V: uv\in E} x_{uv} \le 1 \quad \text{for all } u \in V, \\ & x_e \in \{0,1\} \quad \text{for all } e \in E. \end{aligned} \tag{IP}$$

- In (IP), both the **objective** $\sum_{e\in E} w_e x_e$ and the constraint $\sum_{v\in V:uv\in E} x_{uv}$ are **linear functions** in x.
- Here, a linear function in x is a function of the form

$$c^\top x = \sum_{e\in E} c_e x_e \quad \text{for some } c \in \mathbb{R}^{|E|}.$$

- An optimization problem whose objective and constraints are given by linear functions is called a **linear program**.
- However, the binary constraint in (IP) generates discontinuity and thus cannot be represented by a linear function

Formulating as an optimization problem

$$
\begin{aligned}
\text{maximize} \quad & \sum_{e \in E} w_e x_e \\
\text{subject to} \quad & \sum_{v \in V : uv \in E} x_{uv} \leq 1 \quad \text{for all } u \in V, \\
& x_e \in \{0, 1\} \quad \text{for all } e \in E.
\end{aligned}
\tag{IP}
$$

- However, the binary constraint in (IP) generates discontinuity and thus cannot be represented by a linear function.
- When the objective and constraints except for the binary constraint on its variables are given by linear functions, it is called a **binary linear program**.
- In general, a binary linear program is an **integer linear program** which in general can take any integer-valued variables.
- It is known that integer linear programming and binary linear programming are NP-hard.
- Linear programming admits a **polynomial time** algorithm such as the **ellipsoid method** and the **interior-point algorithm**.

LP relaxation

- A common approach to tackle an integer linear program is to obtain its **linear programming (LP) relaxation**.
- The LP relaxation **relaxes and removes the constraints** to impose that the variables have integer values, and it becomes a linear program.
- The LP relaxation of (IP) is given by

$$
\begin{aligned}
\text{maximize} \quad & \sum_{e \in E} w_e x_e \\
\text{subject to} \quad & \sum_{v \in V : uv \in E} x_{uv} \leq 1 \quad \text{for all } u \in V, \\
& x_e \geq 0 \quad \text{for all } e \in E
\end{aligned}
\tag{LP}
$$

- Any x satisfying the constraints of (LP) would have $x_e \leq 1$ for all $e \in E$, because x_e appears in (degree).

LP relaxation

Lemma

Let $G = (V, E)$ be a graph, not necessarily bipartite, and let $w \in \mathbb{R}^{|E|}$. Then the optimal value of the LP relaxation (LP) is greater than or equal to the optimal value of (IP).

$$
\begin{aligned}
\text{maximize} \quad & \sum_{e \in E} w_e x_e \\
\text{subject to} \quad & \sum_{v \in V : uv \in E} x_{uv} \leq 1 \quad \text{for all } u \in V, \\
& x_e \geq 0 \quad \text{for all } e \in E
\end{aligned}
\tag{LP}
$$

- The linear program (LP) provides an upper bound on the maximum size of a matching in any graph that is not necessarily bipartite.
- However, an optimal solution x^* to (LP) can have fractional parts in $(0, 1)$ in which case x^* does not correspond to a matching.
- **Nevertheless, we can prove that bipartite graphs do not have such an issue.**

Exactness of the LP relaxation for bipartite graphs

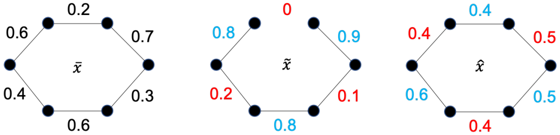
Theorem

Let $G = (V, E)$ be a bipartite graph, and let $w \in \mathbb{R}^{|E|}$. Then the LP relaxation (LP) has an optimal solution x^* that satisfies $x^*_e \in \{0, 1\}$ for all $e \in E$. Moreover, one can find a maximum matching in G by solving the linear program (LP).

- Let $\bar{x} \in [0, 1]^{|E|}$ be an optimal solution to (LP).
- Consider $S = \{e \in E : \bar{x}_e > 0\}$ and the subgraph H of G obtained by deleting the edges that are not in S.
- S may have **cycles** and **large trees**.
- We break the cycles and trees to obtain a matching.

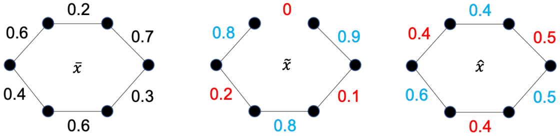
Exactness of the LP relaxation for bipartite graphs

Breaking cycles



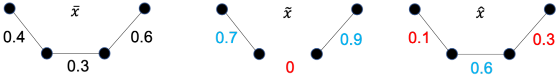
Exactness of the LP relaxation for bipartite graphs

Breaking cycles



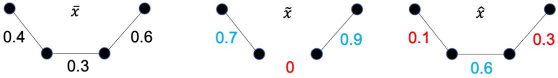
Exactness of the LP relaxation for bipartite graphs

Breaking trees



Exactness of the LP relaxation for bipartite graphs

Breaking trees



LP-based algorithm for maximum weight bipartite matching

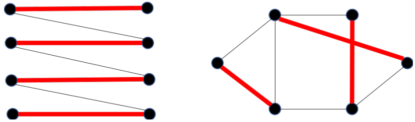
Algorithm 1 LP-based algorithm for maximum weight bipartite matching

Solve the linear program (LP) and get an optimal solution $\bar{x}$
Take $S = \{e \in E : \bar{x}_e > 0\}$ and the corresponding subgraph H
while H contains a cycle **do**
    Find a cycle C in H
    Break C by updating $\bar{x}$ and S
end while
while H contains a tree with at least three vertices **do**
    Take a connected component T of H
    Break T by updating $\bar{x}$ and S
end while
Return S

- In practice, we may use the **simplex method** for solving the LP relaxation (LP).
- For (LP), we can in fact argue that the simplex method directly finds an optimal solution $\bar{x}$ with $\bar{x}_e \in \{0, 1\}$ for all $e \in E$.

Perfect matching

- Given a graph $G = (V, E)$, not necessarily bipartite, a matching M in G is **perfect** if every vertex $v \in V$ is incident to an edge in M.
- In other words, every vertex is attached to a matching edge in a perfect matching.



Maximum weight perfect matching

- We can compute a perfect matching by solving an optimization problem described as follows.
- As in the previous section, we use x_e to indicate whether e is picked for our matching M or not.
- To guarantee that M is a perfect matching, we impose the constraint

$$\sum_{v \in V : uv \in E} x_v = 1$$

  for any vertex $u \in V$.
- Optimization problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} w_e x_e \\ \text{subject to} \quad & \sum_{v \in V : uv \in E} x_{uv} = 1 \quad \text{for all } u \in V, \\ & x_e \in \{0, 1\} \quad \text{for all } e \in E. \end{aligned} \tag{Perfect}$$

Maximum weight perfect matching

- A matching always exists as the empty set is trivially a matching.
- However, aa perfect matching does not always exsits even for a bipartite graph.

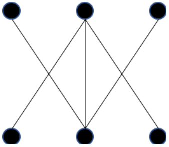


Figure: a bipartite graph that does not admit a perfect matching

Hall's marriage theorem

- This means that the integer linear program (Perfect) may not have a feasible solution.
- For bipartite graphs, we have a simple structural characterization for the existence of a perfect matching.
- The characterization is referred to as **Hall's marriage theorem**.
- Given a subset $S \subseteq V$, $N(S)$ denotes the set of vertices that are adjacent to any vertex in S.

Theorem (Hall's marriage theorem)

Let $G = (V, E)$ be a bipartite graph where V is partitioned into V_1 and V_2. Then G has a perfect matching if and only if $|N(S)| \geq |S|$ for any $S \subseteq V_1$.

Hall's marriage theorem

Theorem (Hall's marriage theorem)

Let $G = (V, E)$ be a bipartite graph where V is partitioned into V_1 and V_2. Then G has a perfect matching if and only if $|N(S)| \geq |S|$ for any $S \subseteq V_1$.

Hall's marriage theorem

Theorem (Hall's marriage theorem)

Let $G = (V, E)$ be a bipartite graph where V is partitioned into V_1 and V_2. Then G has a perfect matching if and only if $|N(S)| \geq |S|$ for any $S \subseteq V_1$.

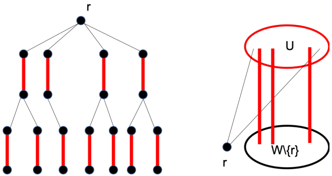


Figure: the M-alternating tree from an M-exposed vertex