Introduction to Modern Combinatorial Optimization **POSTECH, Winter 2025**
Lecture #4: König's theorem and the Hungarian algorithm January 14, 2025
Lecturer: Dabeen Lee

Outline

The first part of this lecture is about the vertex cover problem and König's theorem. We provide a combinatorial proof and an linear programming-based proof for König's theorem. For the second part, we explain the Hungarian algorithm-based method for computin a maximum weight matching in a bipartite graph.

1 Vertex cover problem

So far, we have focused on the bipartite matching problem. Just for a moment, let us turn our attention to a different problem, yet it is closely related to bipartite matching. Given a graph $G = (V, E)$, a subset B of the vertex set V is called a **vertex cover** if for every edge $e \in E$, e has an endpoint in B. The **vertex cover problem** is to find a vertex cover with the minimum

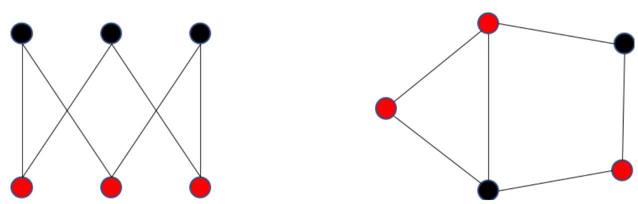


Figure 4.1: vertex cover examples

number of vertices. The following provides a bridge between the matching problem and the vertex cover problem.

Proposition 4.1. *Let $G = (V, E)$ be a graph. Then the minimum size of a vertex cover for G is greater than or equal to the maximum size of a matching in G.*

Proof. Let M be a maximum matching of G. Note that the edges in M are pairwise vertex-disjoint. This means that any vertex cover B contains at least one endpoint of each edge in M, which implies that $|B| \geq |M|$. □

For a bipartite graph, we can derive the folllowing stronger result.

Theorem 4.2 (König's theorem)**.** *Let $G = (V, E)$ be a bipartite graph. Then the minimum size of a vertex cover for G equals the maximum size of a matching in G.*

Proof. Remember the augmenting path algorithm for maximum bipartite matching and the alternating tree procedure to find an M-augmenting path. Suppose that M is a maximum matching in G. Then we know that G has no M-augmenting path. In this case, the alternating tree procedure ends up with a decomposition of the vertex set V into

$$V = (W_1 \cup V_1) \cup \cdots \cup (W_k \cup V_k)$$

for some k illustrated as in Figure 4.2. We proved that every edge $e \in E$ is incident to a vertex in $V_! \cup \cdots \cup V_k$. This means that $V_1 \cup \cdots \cup V_k$ is a vertex cover. Moreover, recall that $|M| = |V_1 \cup \cdots \cup V_k|$.

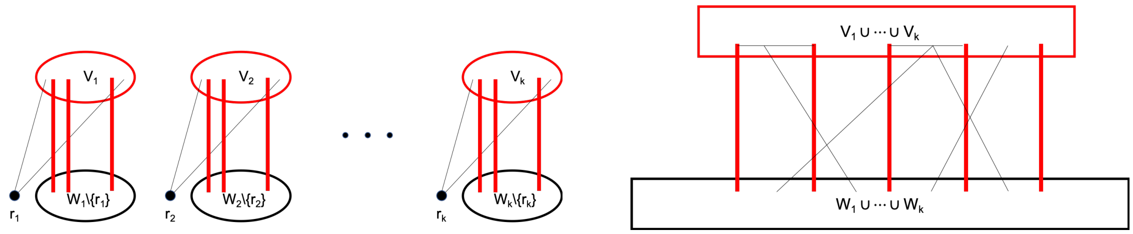


Figure 4.2: vertex set decomposition by the alternating tree procedure

By Proposition 4.1, it follows that $V_1 \cup \cdots \cup V_k$ is a minimum vertex cover and its size equals the maximum size of a matching, as required. □

The proof of Theorem 4.2 suggests that the augmenting path algorithm with the alternating tree procedure not only gives us a maximum matching but also a minimum vertex cover. This means that the vertex cover problem can be solved in polynomial time, but the vertex cover problem for general graphs is known to be NP-hard.

2 Linear programming duality-based proof for König's theorem

As for the matching problem, vertex cover also admits an integer linear programming formulation. For each vertex $v \in V$, we use a variable y_v to indicate whether v is picked for our vertex cover B or not, i.e.,

$$y_v = \begin{cases} 1 & \text{if } v \text{ is included in vertex cover } B, \\ 0 & \text{otherwise.} \end{cases}$$

Then we may impose the condition that y corresponds to a vertex cover by setting

$$y_u + y_v \geq 1$$

for all $uv \in E$. Therefore, the vertex cover problem can be equivalently formulated as the following integer linear program:

$$\begin{aligned} \text{minimize} \quad & \sum_{v \in V} y_v \\ \text{subject to} \quad & y_u + y_v \geq 1 \quad \text{for all } uv \in E, \\ & y_v \in \{0, 1\} \quad \text{for all } v \in V. \end{aligned} \tag{4.1}$$

Proposition 4.3. *Let $G = (V, E)$ be a graph, not necessarily bipartite. Then solving the optimization problem (4.1) computes a minimum vertex cover for G.*

The LP relaxation of (4.1) is given by

$$\begin{aligned} \text{minimize} \quad & \sum_{v \in V} y_v \\ \text{subject to} \quad & y_u + y_v \geq 1 \quad \text{for all } uv \in E, \\ & y_v \geq 0 \quad \text{for all } v \in V. \end{aligned} \tag{4.2}$$

Theorem 4.4. *Let $G = (V, E)$ be a bipartite graph. Then the LP relaxation (4.2) has an optimal solution y^* that satisfies $y^*_v \in \{0, 1\}$ for all $v \in V$. Moreover, one can find a minimum vertex cover for G by solving the linear program (4.2).*

Proof. Let $\bar{y}$ be an optimal solution to (4.2). By the nonnegativity constraint, we have $\bar{y}_v \geq 0$ for all $v \in V$. If $\bar{y}_v > 1$ for some $v \in V$, then one may replace $\bar{y}_v$ with 1 to improve the objective while keeping feasibility. This means that $\bar{y}_v \leq 1$ for all $v \in V$ because $\bar{y}$ is an optimal solution.

Let the vertex V be partitioned into V_1 and V_2. Then we run the following procedure.

1. Pick a random threshold $\theta \in (0,1)$ uniformly at random.
2. Take $U_1 = \{v \in V_1 : \bar{y}_v \geq \theta\}$ and $U_2 = \{v \in V_2 : \bar{y}_v \geq 1-\theta\}$.
3. Define $y^* \in \{0,1\}^{|V|}$ as the incidence vector of $U_1 \cup U_2$.

Let $uv \in E$ with $u \in V_1$ and $v \in V_2$. Note that either $u \in U_1$ or $v \in V_1$ holds, for otherwise, $\bar{y}_u + \bar{y}_v < \theta + (1-\theta) = 1$. This shows that $U_1 \cup U_2$ is a vertex cover. Note that

$$\begin{aligned} \mathbb{E}_\theta\left[\sum_{v\in V} y_v^*\right] &= \sum_{v\in V_1} \mathbb{E}_\theta\left[y_v^*\right] + \sum_{v\in V_2} \mathbb{E}_\theta\left[y_v^*\right] \\ &= \sum_{v\in V_1} \mathbb{P}_\theta\left[\bar{y}_v \geq \theta\right] + \sum_{v\in V_2} \mathbb{P}_\theta\left[\bar{y}_v \geq 1-\theta\right] \\ &= \sum_{v\in V_1} \bar{y}_v + \sum_{v\in V_2} \bar{y}_v \\ &= \sum_{v\in V} \bar{y}_v \end{aligned} \tag{4.3}$$

where the first equality is by the linearity of expectation, the second equality is by the definition of U_1 and U_2, and the third equality holds because θ is chosen uniformly at random.

Recall that y^* under any threshold θ corresponds to a vertex cover, so we have

$$\sum_{v\in V} y_v^* \geq \sum_{v\in V} \bar{y}_v.$$

Then it follows from (4.3) that for any threshold $\theta \in (0,1)$, $y^* \in \{0,1\}^{|V|}$ satisfies

$$\sum_{v\in V} y_v^* = \sum_{v\in V} \bar{y}_v.$$

This in turn implies that y^* for any choice of θ corresponds to a minimum vertex cover. □

Consequently, the optimal value of (4.2) equals that of (4.1) when the graph G is bipartite. Moreover, the proof of Theorem 4.4 provides the following algorithm for computing a minimum vertex cover in a bipartite graph. The proof of Theorem 4.4 guarantees that Algorithm 1 returns a

Algorithm 1 LP-based algorithm for minimum vertex cover

The bipartition $V_1 \cup V_2$ of the vertex set V
Solve the linear program (4.2) and get an optimal solution $\bar{y}$
Take $U_1 = \{v \in V_1 : \bar{y}_v \geq 1/2\}$ and $U_2 = \{v \in V_2 : \bar{y}_v \geq 1/2\}$
Return $U_1 \cup U_2$

minimum vertex cover for a bipartite graph.

Lastly, we conclude this lecture by describing an alternate proof for König's theorem stating that the minimum size of a vertex cover equals the maximum size of a matching in a bipartite graph. Recall that the optimal value of the linear program

$$\begin{aligned} \text{maximize} \quad & \sum_{e\in E} w_e x_e \\ \text{subject to} \quad & \sum_{v\in V: uv\in E} x_{uv} \leq 1 \quad \text{for all } u \in V, \\ & x_e \geq 0 \quad \text{for all } e \in E \end{aligned} \tag{4.4}$$

is equal to the maximum size of a matching. Moreover, we have just proved that the optimal value of the linear program (4.2) is equal to the minimum size of a vertex cover by Theorem 4.4. In fact, the linear program (4.2) is the **linear programming dual** of (4.4). Then by the **strong duality theorem for linear programming**,

$$\begin{aligned} & \min\left\{\sum_{v\in V} y_v : \ y_u + y_v \geq 1 \quad \text{for all } uv \in E, \ y \in \mathbb{R}_+^{|V|}\right\} \\ & = \max\left\{\sum_{e\in E} w_e x_e : \ \sum_{v\in V: uv\in E} x_{uv} \leq 1 \quad \text{for all } u \in V, \ x \in \mathbb{R}_+^{|E|}\right\}. \end{aligned}$$

This leads us to the conclusion that the minimum size of a vertex cover equals the maximum size of a matching in a bipartite graph, which is König's theorem.

3 Hungarian algorithm for maximum weight bipartite matching

In Lecture 3, we learned an LP-based algorithm for computing a maximum weight matching in a bipartite graph. In this section, we introduce a combinatorial algorithm, that is known as the **Hungarian algorithm**.

Preprocessing step Let $G = (V, E)$ be a bipartite graphm and let $w \in \mathbb{R}^{|E|}$ be the edge weight vector.

1. First, as we are interested in a maximum weight matching, we may discard edges with a negative weight.

2. Up to adding dummy vertices and dummy edges with weight zero, we obtain a complete bipartite graph $K_{n,n}$ for some $n \geq 1$.

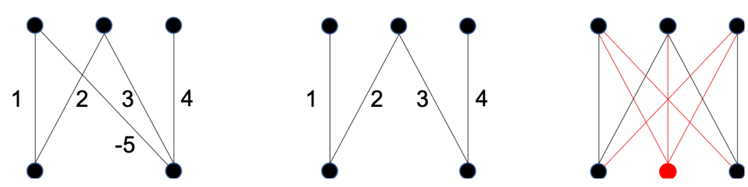


Figure 4.3: illustrating the preprocessing step

We may delete the dummy vertices and dummy edges later.

After the preprocessing step, we may assume that $G = K_{n,n}$ for some $n \geq 1$ and $w \in \mathbb{R}_+^{|E|}$, in which case the problem boils down to finding a **maximum weight perfect matching** in G. As before, let the vertex set V be partitioned into V_1 amd V_2 with $|V_1| = |V_2| = n$. Then a maximum weight matching in G can be computed by

$$\begin{aligned}
\text{maximize} \quad & \sum_{e\in E} w_e x_e \\
\text{subject to} \quad & \sum_{v\in V_2} x_{uv} \leq 1 \quad \text{for all } u \in V_1, \\
& \sum_{u\in V_1} x_{uv} \leq 1 \quad \text{for all } v \in V_2, \\
& x_e \geq 0 \quad \text{for all } e \in E.
\end{aligned} \tag{4.5}$$

Again, as $w_e \geq 0$ for all $e \in E$ and G is a complete bipartite graph, (4.6) has an optimal solution that corresponds to a perfect matching. Then it follows that (4.6) is equivalent to

$$\begin{aligned}
\text{maximize} \quad & \sum_{e\in E} w_e x_e \\
\text{subject to} \quad & \sum_{v\in V_2} x_{uv} = 1 \quad \text{for all } u \in V_1, \\
& \sum_{u\in V_1} x_{uv} = 1 \quad \text{for all } v \in V_2, \\
& x_e \geq 0 \quad \text{for all } e \in E.
\end{aligned} \tag{4.6}$$

The dual of (4.6) is given by

$$\begin{aligned}
\text{minimize} \quad & \sum_{u\in V_1} y_u + \sum_{v\in V_2} z_v \\
\text{subject to} \quad & y_u + z_v \geq w_{uv} \quad \text{for all } uv \in E.
\end{aligned} \tag{4.7}$$

The following result is a direct consequence of the **complementary slackness condition** for linear programming, while we state its direct proof.

Lemma 4.5. *Let M be a perfect matching in G. Suppose that there exists a feasible solution (y, z) to (4.7) that satisfies $y_u + z_v = w_{uv}$ for every $uv \in M$. Then M is a maximum weight matching.*

Proof. Let M' be a perfect matching in G, and let (y', z') be a solution satisfying the constraints of (4.7). Then Note that for any solution (y, z) satisfying the constraints of (4.7), we have

$$\sum_{uv\in M'} w_{uv} \leq \sum_{uv\in M'} (y'_u + z'_v) = \sum_{u\in V_1} y'_u + \sum_{v\in V_2} z'_v$$

where the equality holds because M' is a perfect matching. This implies that

$$\begin{aligned}
&\max\left\{ \sum_{uv\in M'} w_{uv} : \ M' \text{ is a perfect matching} \right\} \\
&\leq \min\left\{ \sum_{u\in V_1} y'_u + \sum_{v\in V_2} z'_v : \ y'_u + z'_v \geq w_{uv} \quad \text{for all } uv \in E \right\}
\end{aligned}$$

If some (y, z) satisfies $y_u + z_v = w_{uv}$ for every $uv \in M$, then it follows that

$$\sum_{uv\in M} w_{uv} = \sum_{uv\in M} (\bar{y}_u + \bar{z}_v) = \sum_{u\in V_1} \bar{y}_u + \sum_{v\in V_2} \bar{z}_v.$$

This indicates that the weight of M achieves the maximum possible, and therefore, M is a maximum weight matching. $\square$

Based on Lemma 4.5, the main idea behind the Hungarian algorithm is as follows.

- (y, z) always remains feasible to (4.7), satisfying the constraints of (4.7).
- Only an edge $uv \in E$ satisfying $y_u + z_v = w_{uv}$ can be added to our matching M.

Once our matching M becomes a perfect matching, then it will satisfy the conditions of Lemma 4.5, which guarantees that M is a maximum weight matching. To implement this idea, we introduce the notion of **equality subgraphs**. Given a feasible solution (y, z) to (4.7), we define the subgraph of G taking the edges $uv \in E$ satisfying $y_u + z_v = w_{uv}$. We use notation $G_{y,z}$ to denote the equality subgraph of G associated with (y, z).

- Given a feasible solution (y, z) to (4.7), we take a maximum matching M in $G_{y,z}$.

Based on this, we deduce the Hungarian algorithm.

Algorithm 2 Hungarian algorithm for maximum weight bipartite matching

Input: complete bipartite graph $G = (V, E)$ with $V = V_1 \cup V_2$ and $w \in \mathbb{R}_+^{|E|}$
Initialize $y_u = \max_{v\in V_2} w_{uv}$ for $u \in V_1$, $z_v = 0$ for $v \in V_2$
Initialize $M = \emptyset$ and $B = \emptyset$
while M is not a perfect matching **do**
  Construct the equality subgraph $G_{y,z}$ associated with (y, z)
  Set M and B as a maximum matching and a minimum vertex cover in $G_{y,z}$, respectively
  Set $R = V_1 \cap B$ and $T = V_2 \cap B$
  Compute $\epsilon = \min\{y_u + z_v - w_{uv} : u \in V_1 - R, v \in V_2 - T\}$
  Update $y_u = y_u - \epsilon$ for $u \in V_1 - R$ and $z_v = z_v + \epsilon$ for $v \in T$
end while
Return M

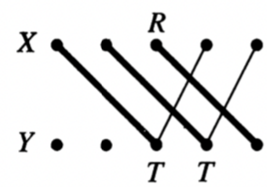
Example 4.6 (Example 3.2.10., West)**.** Let us consider an example with $G = K_{5,5}$ given by Figure 4.4. In each matrix, the rows correspond to the vertices in V_1, and the columns are for the vertices in V_2.

Theorem 4.7. *Let $G = (V, E)$ be a complete bipartite graph, and let $w \in \mathbb{R}_+^{|E|}$. Then Algorithm 2 finds a maximum weight pefect matching in G.*

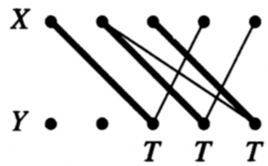
Proof. First, we argue that (y, z) always remains feasible to (4.7). The initial solution with $y_u = \max_{v\in V_2} w_{uv}$ for $u \in V_1$, $z_v = 0$ for $v \in V_2$ is feasible because $\max_{v\in V_2} w_{uv} \geq w_{uv}$ for any $uv \in E$. Suppose that (y, z) is feasible to (4.7) at some point of running Algorithm 2. Let M and B be a maximum matching and a minimum vertex cover in $G_{y,z}$, respectively. Moreover, we take $R = V_1 \cap B$ and $T = V_2 \cap B$ and

$$\epsilon = \min\{y_u + z_v - w_{uv} :\ u \in V_1 - R,\ v \in V_2 - T\}.$$

$$
\begin{pmatrix} 4 & 1 & 6 & 2 & 3 \\ 5 & 0 & 3 & 7 & 6 \\ 2 & 3 & 4 & 5 & 8 \\ 3 & 4 & 6 & 3 & 4 \\ 4 & 6 & 5 & 8 & 6 \end{pmatrix}
\rightarrow
\begin{array}{c c c}
 & \begin{matrix} 0 & 0 & 0 & 0 & 0 \end{matrix} & \\
\begin{matrix} 6 \\ 7 \\ 8 \\ 6 \\ 8 \end{matrix} & \begin{pmatrix} 2 & 5 & \underline{0} & 4 & 3 \\ 2 & 7 & 4 & \underline{0} & 1 \\ 6 & 5 & 4 & 3 & \underline{0} \\ 3 & 2 & 0 & 3 & 2 \\ 4 & 2 & 3 & 0 & 2 \end{pmatrix} & \begin{matrix} \\ \\ R \\ \\ \\ \end{matrix} \\
 & \begin{matrix} & & T & T & \end{matrix} &
\end{array}
$$



$$
\begin{array}{c c}
 & \begin{matrix} 0 & 0 & 1 & 1 & 0 \end{matrix} \\
\begin{matrix} 5 \\ 6 \\ 8 \\ 5 \\ 7 \end{matrix} & \begin{pmatrix} 1 & 4 & \underline{0} & 4 & 2 \\ 1 & 6 & 4 & \underline{0} & 0 \\ 6 & 5 & 5 & 4 & \underline{0} \\ 2 & 1 & 0 & 3 & 1 \\ 3 & 1 & 3 & 0 & 1 \end{pmatrix} \\
 & \begin{matrix} & & T & T & T \end{matrix}
\end{array}
$$



$$
\rightarrow
\begin{array}{c c}
 & \begin{matrix} 0 & 0 & 2 & 2 & 1 \end{matrix} \\
\begin{matrix} 4 \\ 5 \\ 7 \\ 4 \\ 6 \end{matrix} & \begin{pmatrix} 0 & 3 & \underline{0} & 4 & 2 \\ \underline{0} & 5 & 4 & 0 & 0 \\ 5 & 4 & 5 & 4 & \underline{0} \\ 1 & \underline{0} & 0 & 3 & 1 \\ 2 & 0 & 3 & \underline{0} & 1 \end{pmatrix}
\end{array}
$$

Figure 4.4: an example of running the Hungarian algorithm

Assume that M is not a perfect matching. Then. let (y', z') denote what is obtained from (y, z) after the update. It is sufficient to check that $y'_u + z'_v \geq w_{uv}$ for $u \in V_1 - R$ and $v \in V_2$. Note that for $u \in V_1 - R$,

$$y'_u + z'_v = \begin{cases} y_u - \epsilon + z_v & \text{if } v \in V_2 - T, \\ y_u - \epsilon + z_v + \epsilon & \text{if } v \in T. \end{cases}$$

Moreover, for $u \in V_1 - R$ and $v \in V_2 - T$, we have

$$y_u + z_v - \epsilon \geq y_u + z_v - (y_u + z_v - w_{uv}) = w_{uv}.$$

Therefore, what remains is to argue that Algorithm 2 terminates with a perfect matching. Suppose that the current matching M is not a pefect matching, in which case $B = R \cup T$ is not a vertex cover of G. That meanas that there exists an edge not covered by B, which implies that the has not yet appeared in the equality subgraph $G_{y,z}$. Therefore, we must have $\epsilon > 0$. Let $u \in V_1 - R$ and $v \in V_2 - T$ be such that $y_u + z_v - w_{uv} = \epsilon$. Then after the update, we have

$$y'_u + z'_v - w_{uv} = y_u - \epsilon + z_v - w_{uv} = 0.$$

Hence, the edge uv newly enters the equality subgraph. That said, one instance of the while loop increases the number of edges in the equality subgraph by at least 1. Note that G has $O(|V|^2)$ edges in total, so the algorithm will terminate eventually. $\square$