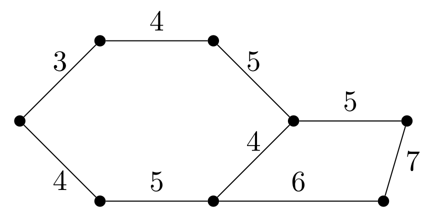
Fall 2016, MATH-566

# Minimum-Weight Perfect Matching in Bipartite Graphs

Source: Bill,Bill,Bill

Let $G = (V, E)$ be a graph. Let $c : E \to \mathbb{R}^+$ is a cost. Find a perfect matching $M$ that is minimizing the sum of costs of the edges in the matching.

**1:** Find minimum-weight perfect matching in the following graph:




**2:** Write the minimum-weight perfect matching as an integer program $(IP)$ on a graph $G = (V, E)$.

**Solution:**

$$(IP)\begin{cases}\text{minimize} & \sum_{e\in E} c(e)x_e\\ \text{subject to} & \sum_{e\in\delta(v)} x_e = 1 \text{ for all } v\in V\\ & \mathbf{x}\in\{0,1\}^{|E|},\end{cases}$$

**3:** Consider a relaxation of $(IP)$ to a linear program $(P)$ and write the dual $(D)$ of $(P)$.

**Solution:**

$$(P)\begin{cases}\text{minimize} & \sum_{e\in E} c(e)x_e\\ \text{subject to} & \sum_{e\in\delta(v)} x_e = 1 \text{ for all } v\in V\\ & x_e \ge 0 \text{ for all } e\in E\end{cases}$$

$$(D)\begin{cases}\text{maximize} & \sum_{v\in V} y_v\\ \text{subject to} & y_u + y_v \le c(uv) \text{ for all } uv\in E\\ & y_v\in\mathbb{R} \text{ for all } v\in V\end{cases}$$

**Theorem** Birkhoff: If $G$ is a bipartite graph, then solution to $(P)$ is integral. (Why?)
Use minimum cost flow.

**4:** Formulate complementary slackness conditions for optimal solution $\mathbf{x}$ of $(P)$ and optimal solution $\mathbf{y}$ of $(D)$.

**Solution:**
If $x_e > 0$, then $y_u + y_v = c(uv)$.
If $y_u + y_v < c(uv)$ then $x_e = 0$, then.

Algorithm idea: Maintain an optimal solution to $(D)$, create a solution to $(P)$ whose value is matching the dual solution.

**5:** Find initial solutions to $(P)$ and $(D)$, where solution to $(D)$ is feasible and the solutions satisfy complementary slackness. (Solution to $(P)$ does not have to be feasible.)

**Solution:** $\mathbf{x} = 0$, $\mathbf{y} = 0$ will do.

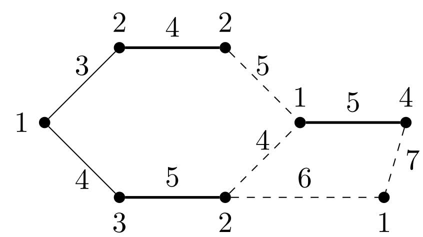
**6:** If the solution to $(D)$ is fixed, which edges can be used in matching? (Denote the edges by $E_=$.)

**Solution:** All edges $e = uv$, where $y_u + y_v = c(e)$ can have $x_e > 0$.

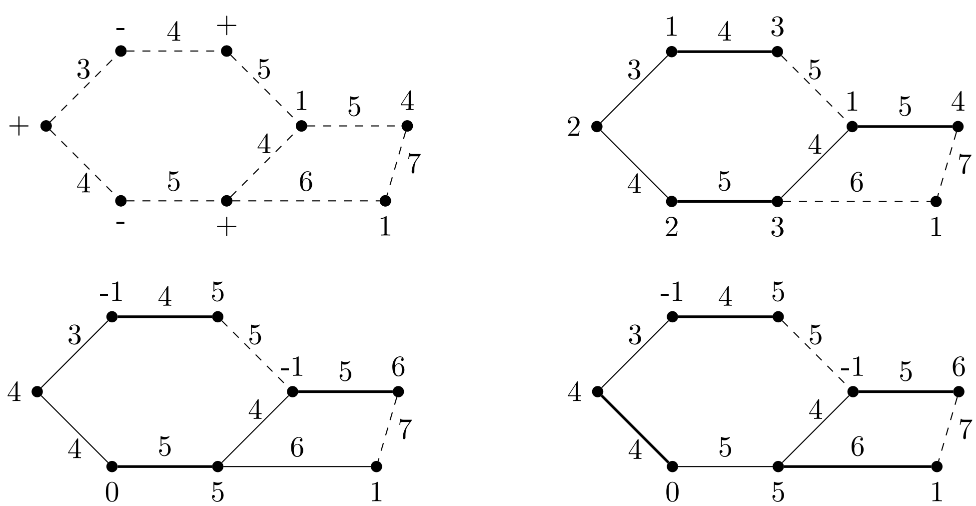
Algorithm sketch, suppose a perfect matching exists.

- start with initial solution $\mathbf{x}$,$\mathbf{y}$.
- Take edges $E_=$ and try to find a perfect matching $M$ by growing augmenting forests
- If $M$ is not perfect, there are some outer of $F$ vertices adjacent to edges in $E$ but not in $E_=$.
- Update $\mathbf{y}$ to allow more edges in $E_=$ and repeat.

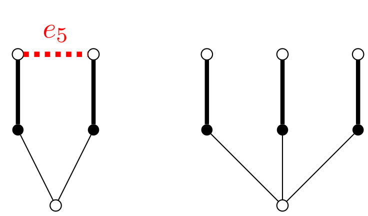
**7:** What to do if there is no perfect matching in $E_=$? Consider the following example. Number on edge $e$ is $c(e)$, number at vertex $v$ is $y_v$. How to modify $\mathbf{y}$ to allow the tree to grow?



**Solution:**



Recall that during growing the tree, we encountered edges like $e_5$ that were not possible to drop from the tree. The above algorithm does not work for edges with $e_5$.




Our algorithm works only for **bipartite** graphs. Can be (nontrivially) generalized for all graphs. (The linear program has to be stronger by adding more constraints - for all odd vertex subsets, at least one edge is in the cut, blossoms need to be treated carefully.)