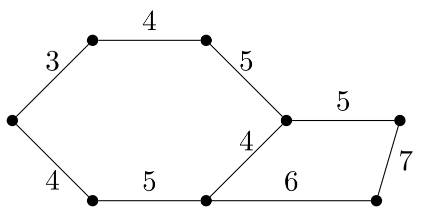
Minimum-Weight Perfect Matching in Bipartite Graphs

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. Works ONLY for bipartite graphs. For general graphs, this can have values also $\frac{1}{2}$.

4: Formulate complementary slackness conditions for an optimal solutions $\mathbf{x}$ of (P) and $\mathbf{y}$ 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 a feasible solution to (D), create a solution to (P) whose value is matching the dual solution.

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

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

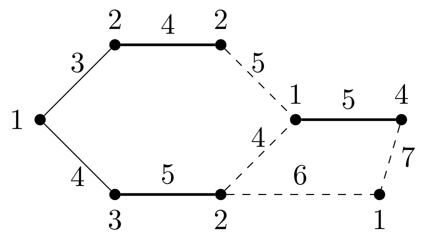
6: If the solution to (D) is fixed, which edges can be used in a matching corresponding to (P)? I.e. which x_e from (P) can be 1 while maintaining complementary slackness? (Denote these edges by $E_=$.)

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

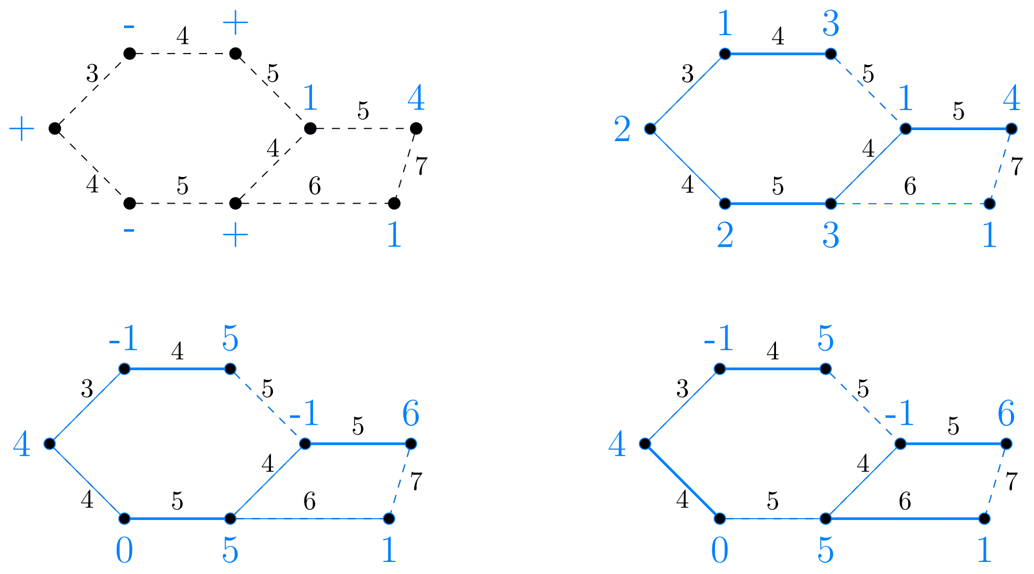
Algorithm sketch, suppose a perfect matching exists.

- start with an initial solution **x**,**y**.
- Take edges $E_=$ and try to find a perfect matching M by growing augmenting forest F.
- If M is not perfect, there are some outer vertices of F adjacent to edges in E but not in $E_=$.
- Update **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. A number on an edge e is $c(e)$, a number at vertex v is y_v. How to modify **y** to allow the tree to grow? Try (in steps)



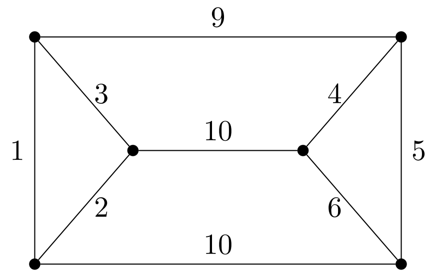
Solution:



8: Recall the (P) and (D) are

$$(P)\begin{cases}\text{minimize} & \sum_{e\in E} c(e)x_e\\ \text{subject to} & \sum_{e\in\delta(v)} x_e = 1 \text{ for all } v\in V\\ & x_e \geq 0 \text{ for all } e\in E\end{cases}\qquad (D)\begin{cases}\text{maximize} & \sum_{v\in V} y_v\\ \text{subject to} & y_u+y_v \leq c(uv) \text{ for all } uv\in E\\ & y_v\in\mathbb{R} \text{ for all } v\in V\end{cases}$$

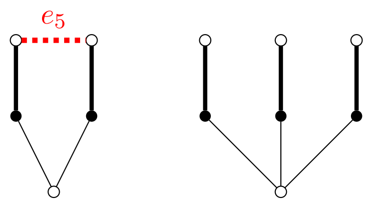
Show that the optimal solution for (P) is not integral on the following graph and find what is the minimum-weight perfect matching for it.



Solution: Optimal solution of (P) is $x_e = 1/2$ for edges in triangles. This gives cost 10.5. On the other hand, the minimum-weight perfect matching is has weight 16, which comes from edges of weight 1,5, and one edge of weight 10 between.

Recall that during growing the alternating forest, we encountered edges like e_5 that were not possible to ignore and we created blossoms when we found these.

9: Why the algorithm for bipartite graphs does not work for edges like e_5?



Solution: Because in the dual solution, the y for both endpoints of e_5 is changing by the same amount. Hence the difference is always the same and the edge cannot be added to $E_=$.

Our algorithm works only for **bipartite** graphs. The algorithm can be (nontrivially) generalized for all graphs by using blossoms. The linear program has to be stronger by adding more constraints - for all odd vertex subsets, at least one edge is in the cut as we will see next time.