1 Optimality Conditions

In the previous lecture, we defined the *minimum cost perfect bipartite matching* problem (also known as the assignment problem): We are given a bipartite graph $G = (V \cup W, E)$ where each edge $e \in E$ is associated with a cost $c_e \geq 0$. Our goal is to output a perfect matching $M \subseteq E$ that minimizes the cost $\sum_{e \in M} c_e$.

Before we describe the algorithm for finding a minimum cost matching, we look for optimality conditions: a way to “certify” that a given perfect matching has minimum cost. For that, the following definition will be useful.

**Definition 1.1.** Let $G = (V \cup W, E)$ be a bipartite graph and $M \subseteq E$ be a matching (not necessarily a perfect matching). A cycle $C$ in $G$ is called an $M$-alternating cycle if the edges of the cycle alternate between $M$ and $E \setminus M$ (each edge from $M$ in the cycle is followed by an edge not in $M$, and vice versa).

We define the cost of an $M$-alternating cycle $C$ as $\text{cost}_M(C) = \sum_{e \in C \setminus M} c_e - \sum_{e \in C \cap M} c_e$. An $M$-alternating cycle $C$ is called negative if $\text{cost}_M(C) < 0$ and nonnegative otherwise.

Note that alternating cycles and their costs are defined with respect to a specific matching $M$. In particular, when we define the cost of a cycle, the edge not in $M$ are counted with a positive sign, and the edges of $M$ get a negative sign. The reason for that is, given an $M$-alternating cycle, we can “toggle” its edges — remove each edge in $C \cap M$ from the matching and add each edge in $C \setminus M$ and get a new matching $M'$. $M'$ is going to be a valid matching for exactly the same set of vertices that are matched in $M$ (why?). The cost of the new matching $M'$ is going to be the cost of $M$ plus the cost of the $M$-alternating cycle $\text{cost}_M(C)$. Similarly, given $M'$, we can toggle the edges of the same cycle $C$ (which is also $M'$-alternating), pay $\text{cost}_{M'}(C) = -\text{cost}_M(C)$, and get the matching $M$.

We are now ready to show that a perfect matching has minimum cost if and only if there are no $M$-alternating negative cycles. This will be the optimality condition for the minimum cost bipartite matching.

---

*The discussion of optimality conditions mostly follows [2]. The discussion of the algorithm mostly follows [1] and [3].

1 We assume that $|V| = |W|$ and that the graph admits a perfect matching. If $|V| \neq |W|$, dummy nodes with no edges can be added on the smaller side. If the graph admits no perfect matching, we can either detect that (and return that there is no perfect matching), or add edges with cost $\infty$ between every pair $(v, w)$ such that $(v, w) \notin E$ (this will result in a minimum-cost maximum-cardinality matching).

2 Throughout this lecture, we use $C$ to refer to both the cycle itself and the edge of the cycle.
matching problem: our algorithm will terminate with a matching that has no negative cycles, and this way we will know that it outputs a minimum cost matching.

**Theorem 1.2.** A perfect matching $M$ in a bipartite graph has minimum cost if and only if there are no negative $M$-alternating cycles.

**Proof.** In one direction, we show that if there is a negative cycle, $M$ is not minimum cost. Let $C$ be a negative cycle, and toggle the edges in $C$ to get a new matching $M'$. $M'$ is going to be a perfect matching such that

$$cost(M') = cost(M) + cost_M(C) < cost(M)$$

(since $C$ is a negative $M$-alternating cycle). Hence, $M$ is not minimum cost.

In the other direction, we show that if all cycles are non-negative, $M$ has minimum cost. Let $M'$ be any other perfect matching. We will prove that $cost(M') \geq cost(M)$. Consider $M \oplus M'$, the set of edges that belong to $M$ or $M'$ but not to both. If we consider only the edges in $M \oplus M'$, each vertex has degree either 0 (if it is matched to the vertex in $M$ and $M'$) or 2 (if it is matched to a different vertex in each of $M$ and $M'$). Hence, the set of edges $M \oplus M'$ is a set of disjoint cycles (why?). Each of these cycles is going to an $M$-alternating cycle. Denote this set of $M$-alternating cycles by $\mathcal{C}$. Since we assumed that there are no negative $M$-alternating cycles, for every $C \in \mathcal{C}$, $cost_M(C) \geq 0$. If we start from $M$, and toggle all the cycles in $\mathcal{C}$, we will get $M'$ (note that by toggling the cycles, we remove all the edges $M \setminus M'$ and add all the edges $M' \setminus M$). Furthermore, from the discussion above,

$$cost(M') = cost(M) + \sum_{C \in \mathcal{C}} cost_M(C) \geq cost(M).$$

We conclude that $M$ is a minimum cost bipartite matching. \qed

# 2 Algorithm

We now describe an algorithm for the minimum cost bipartite matching. It is a slightly simplified version of the Hungarian/Kuhn-Munkres algorithm. For the story behind the algorithm, refer to [2].

Following our discussion of optimality conditions, our goal is to find an algorithm that returns a perfect matching with no negative cycles. The algorithm will start with an empty matching and iteratively augment the current matching, such that at each step the matching size increases by one, and there are no negative cycles.

## 2.1 Node Prices

Given a matching, how can we check that there are no negative cycles? The main tool that we are going to use is node prices. We will assign each node $x \in V \cup W$ a price $p(x) \in \mathbb{R}$, and shift the edge costs according to the prices. Using the new edge costs, it will be easy to argue that there are no negative cycles.

The intuition behind node prices is the following: For any node $x$, if we add (or subtract) the same constant $p(x)$ to the cost of all the edges that are adjacent to $x$, the minimum cost perfect
matching will stay the same. The reason is that any perfect matching must include exactly one edge that is adjacent to \( x \). So after we add \( p(x) \) to the edge costs, the cost of any perfect matching \( M \) will be \( \text{cost}(M) + p(x) \). Since we only added a constant to the costs of all matchings, a matching has minimum cost before adding \( p(x) \) if and only if it has minimum cost after adding \( p(x) \). Our goal will be to find prices \( p(x) \) such that after shifting the edge costs, the cost of the matching will be 0 while all the edge costs remain nonnegative. Then it will be easy to argue that there are no negative cycles (Lemma 2.2 below).

Formally, given node prices \( p \), we define the reduced cost \( c^p_{(v,w)} \) of an edge \((v,w)\) to be \( c_{(v,w)} + p(v) - p(w) \), where the edge \((v,w)\) connects \( v \in V \) with \( w \in W \). In general, the edges are undirected, but since it is a bipartite graph, they always connect a node from \( V \) to a node in \( W \), and we treat these nodes asymmetrically: we add the price \( p(v) \) of \( v \in V \) to the edges adjacent to \( v \), and subtract the price \( p(w) \) of \( w \in W \) to all the edges adjacent to \( w \). (This is an arbitrary choice, but we need to be consistent.) At each step, the algorithm will be required to find a set of prices with special properties, which we call compatible prices.

**Definition 2.1.** Let \( M \) be a (not necessarily perfect) matching in a bipartite graph \( G = (V \cup W, E) \). We say that a vector of node prices \( p \) is a set of compatible prices with respect to \( M \) if the two following conditions both hold:

1. For every \((v,w) \in E\), the reduced cost \( c^p_{(v,w)} \) of \((v,w)\) is nonnegative, that is, \( c_{(v,w)} + p(v) - p(w) \geq 0 \).
2. For every \((v,w) \in M\), \( c^p_{(v,w)} = 0 \).

**Lemma 2.2.** Let \( M \) be a matching and \( p \) be compatible prices with respect to \( M \). Then there are no negative \( M \)-alternating cycles.

**Proof.** Let \( C \) be an \( M \)-alternating cycle. We first show that the cost of \( C \) using the original edge costs \( c_e \) is the same as the cost of \( C \) using the reduced edge costs. Denote by \( V(C) \) the set of vertices from \( V \) that appear in \( C \) and by \( W(C) \) the set of vertices from \( W \) that appear in \( C \). Note that each vertex from \( V(C) \) or \( W(C) \) is incident on exactly one edge in \( C \cap M \) and exactly one edge in \( C \backslash M \) (since any vertex is matched once in \( C \cap M \) and \( C \backslash M \)). Then,

\[
\sum_{(v,w) \in C \backslash M} c^p_{(v,w)} - \sum_{(v,w) \in C \cap M} c^p_{(v,w)} = \sum_{(v,w) \in C \backslash M} (c_{(v,w)} + p(v) - p(w))
- \sum_{(v,w) \in C \cap M} (c_{(v,w)} + p(v) - p(w))
= \sum_{(v,w) \in C \backslash M} c_{(v,w)} + \sum_{v \in V(C)} p(v) - \sum_{w \in W(C)} p(w)
- \left( \sum_{(v,w) \in C \cap M} c_{(v,w)} + \sum_{v \in V(C)} p(v) - \sum_{w \in W(C)} p(w) \right)
= \sum_{(v,w) \in C \backslash M} c_{(v,w)} - \sum_{(v,w) \in C \cap M} c_{(v,w)}.
\]

Now, since \( p \) are compatible prices, for any \((v,w) \in M\), \( c^p_{(v,w)} = 0 \). Hence, \( \sum_{(v,w) \in C \backslash M} c^p_{(v,w)} = 0 \). Additionally, the reduced cost of any edge \((v,w)\) is nonnegative, and we get the cost of \( C \) using the
original costs is
\[
\sum_{(v, w) \in C \setminus M} c_{(v, w)} - \sum_{(v, w) \in C \cap M} c_{(v, w)} = \sum_{(v, w) \in C \setminus M} c^p_{(v, w)} - \sum_{(v, w) \in C \cap M} c^p_{(v, w)} = \sum_{(v, w) \in C \setminus M} c^p_{(v, w)} \geq 0.
\]

We conclude that \( C \) is nonnegative, and that there are no negative cycles.

\[\Box\]

### 2.2 Algorithm Description

Lemma 2.2 implies that if at each step, the algorithm can find a matching and a set of compatible prices, we know that there are no negative cycles. The algorithm starts with an empty matching, for which the prices \( p(v) = 0 \) for every \( v \in V \cup W \) are compatible prices. Then, at each step, we will augment the matching using an \( s-t \) path in the residual graph \( G_M \):

- The vertices of \( G_M \) are \( V \cup W \cup \{s, t\} \).
- For every edge \((v, w) \in M\), add a directed edge \((w, v)\) to \( G_M \).
- For every edge \((v, w) \in E \setminus M\), add a directed edge \((v, w)\) to \( G_M \).
- For any unmatched \( v \in V\), add a directed edge \((s, v)\) and for any unmatched \( w \in W\), add \((w, t)\).

This is exactly the residual graph of the flow network we used in the reduction from maximum cardinality bipartite matching to maximum flow (omitting the edges into \( s \) and out of \( t \)). If we find an \( s-t \) path in \( G_M \), we can augment the flow/matching in the same way we did for maximum cardinality matching, and the number of matched vertices will increase by one.

However, since our algorithm needs to also consider the edge costs, we will need to pick a specific \( s-t \) path (instead of any \( s-t \) path). To do that, each edge \((v, w)\) (from \( V \) to \( W \)) in \( G_M \) will be associated with its reduced cost \( c^p_{(v, w)} \) (with respect to the current node prices \( p \)). Edges \((w, v)\) (from \( W \) to \( V \)) will be associated with the cost \(-c^p_{(v, w)}\), and edges from \( s \) or into \( t \) will have cost 0. The algorithm will pick the shortest path from \( s \) to \( t \) (with respect to the above edge costs, not the number of hops). Note that by Definition 2.1, all the edges from \( V \) to \( W \) have nonnegative costs, and the rest of the edges have cost 0. This allows us to use Dijkstra’s algorithm to find the shortest path efficiently.

The full algorithm appears as Algorithm 1.

### 2.3 Analysis

After \(|V|\) iterations, Algorithm 1 terminates with a perfect matching. To complete the analysis, we need to show that the output matching has minimum cost and to analyze the run time.

Regarding the run time, there are \(|V| = O(n)|\) iterations. In each iteration, the expensive part is running Dijkstra’s algorithm (the rest can be done in linear time). Dijkstra’s algorithm can be implemented to run in time \(O(m + n \log n)\), hence the total run time of Algorithm 1 is \(O(mn + n^2 \log n)\).
We need to show that for every edge \((v, w)\) in the matching \(M\), it follows that there are no negative cycles and the matching has minimum cost.

**Proof.** We recall that at the initialization step, the all-zero price vector is compatible with the empty matching.

To argue that the output matching has minimum cost, we will prove that the prices computed at the end of each iteration are compatible prices. Then, from Lemma 2.2 and Theorem 1.2, it follows that there are no negative cycles and the matching has minimum cost.

**Lemma 2.3.** Let \(M\) be a matching and \(p\) be a set of compatible prices at the beginning of an iteration in Algorithm 1. Denote by \(M'\) and \(p'\) the matching and price vector (respectively) at the end of the iteration. Then, \(p'\) is a vector of compatible prices with respect to \(M'\).

**Proof.** We need to show that for every edge \((v, w)\) in \(E\), \(c_{(v, w)}' \geq 0\) and for every \((v, w)\) in \(M'\), \(c_{(v, w)}' = 0\). Consider an edge \((v, w)\) in \(E\) (where \(v \in V\) and \(w \in W\)). It holds that

\[
c_{(v, w)}' = c_{(v, w)} + p'(v) - p'(w) = c_{(v, w)} + p(v) + d(v) - p(w) - d(w) = c_{(v, w)} + d(v) - d(w)
\]

where \(d\) is the shortest path distance vector computed during this iteration. We split the proof into three cases.

1. \((v, w) \in M\): Since the edge belongs to the matching, there is only one edge that enters \(v\) in \(G_M\), which is the edge \((v, v)\) with cost \(-c_{(v, w)}^p\). Then the shortest path from \(s\) to \(v\) has to go through \(w\) and then \((w, v)\). Hence, \(d(v) = d(w) - c_{(v, w)}^p\), and

\[
c_{(v, w)}' = c_{(v, w)} + d(v) - d(w) = c_{(v, w)} + d(w) - c_{(v, w)} = 0.
\]

2. \((v, w) \in M' \setminus M\): Since \((v, w)\) was added to the matching \(M'\) in this iteration, we know that it is along the shortest path from \(s\) to \(t\), and that it is directed in \(G_M\) from \(V\) to \(W\). Then, the shortest path from \(s\) to \(w\) goes through \(v\) and then \((v, w)\). Hence, \(d(w) = d(v) + c_{(v, w)}^p\), and

\[
c_{(v, w)}' = c_{(v, w)} + d(v) - d(w) = c_{(v, w)} + d(v) - (d(v) + c_{(v, w)}^p) = 0.
\]

\footnote{Recall that at the initialization step, the all-zero price vector is compatible with the empty matching.}

---

**Algorithm 1** Minimum Cost Perfect Bipartite Matching

**Input:** bipartite graph \(G = (V \cup W, E)\) with nonnegative edge costs \(c_e\)

**Output:** a minimum cost perfect matching

1. Initialize: set \(M \leftarrow \emptyset\) (initial matching) and \(p(v) \leftarrow 0\) for all \(v \in V \cup W\).

2. While \(M\) is not a perfect matching:

   a. Compute the residual graph \(G_M\).
   b. Use Dijkstra’s algorithm to compute a vector \(d\) of the shortest path distance from \(s\) to every \(v \in V \cup W \cup \{s, t\}\) (using the edge costs described above).
   c. Augment the matching along the shortest \(s-t\) path.
   d. Update the node prices: for every \(v \in V \cup W\), set \(p(v) \leftarrow p(v) + d(v)\).
3. $(v, w) \notin M' \cup M$: Since $(v, w) \notin M$, we know that it is directed from $V$ to $W$ in $G_M$. We only need to show that the reduced cost is nonnegative. The cost of the shortest path from $s$ to $w$ is at most the cost of taking the shortest path from $s$ to $v$ and then $(v, w)$. Hence, $d(w) \leq d(v) + c^p_{(v,w)}$, and
\[
c^p'_{(v,w)} = c^p_{(v,w)} + d(v) - d(w) \geq d(w) - d(w) = 0.
\]
This completes the proof and the analysis of the algorithm.  

References

