22 Matching

Most of us are familiar with the difficult problem of finding a good match. We use graphs to study the problem from a global perspective.

Marriage problem. Suppose there are n boys and n girls and we have a like-dislike relation between them. Representing this data in a square matrix, as in Figure 30 on the left, we write an '×' whenever the corresponding boy and girl like each other. Alternatively, we may represent the data in form of a graph in which we draw an edge for each '×' in the matrix; see Figure 30 on the right. This graph, $G = (V, E)$, is *bipartite*, that is, we can partition the vertex set as $V = X \dot\cup Y$ such that each edge connects a vertex in X with a vertex in Y. The sets X and Y are the *parts* of the graph.

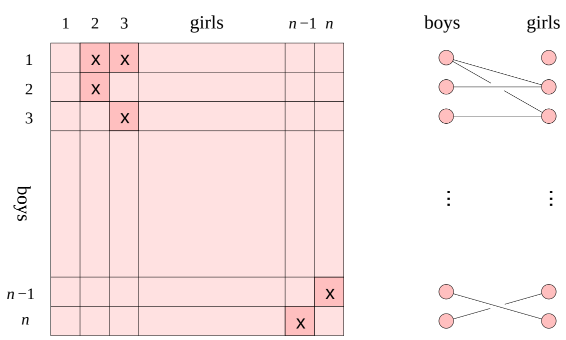

Figure 30: The matrix on the left and the bipartite graph on the right both represent the same data.

The goal is to marry off the boys and girls based on the relation. We formalize this using the bipartite graph representation. A *matching* is a set $M \subseteq E$ of vertex-disjoint edges. The matching is *maximal* if no matching properly contains M. The matching is *maximum* if no matching has more edges than M. Note that every maximum matching is maximal but not the other way round. Maximal matchings are easy to find, eg. by greedily adding one edge at a time. To construct a maximum matching efficiently, we need to know more about the structure of matchings.

Augmenting paths. Let $G = (V, E)$ be a bipartite graph with parts X and Y and $M \subseteq E$ a matching. An *alternating path* alternates between edges in M and edges in $E - M$. An *augmenting path* is an alternating path that begins and ends at unmatched vertices, that is, at vertices that are not incident to edges in M. If we have an augmenting path, we can switch its edges to increase the size of the matching, as in Figure 31.

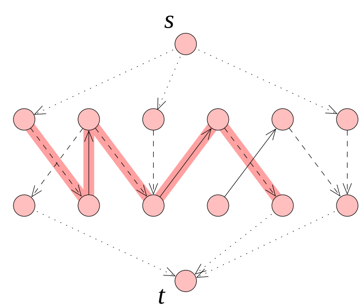

Figure 31: The solid edges form a matching. The shaded edges form an augmenting path. Trading its dashed for its solid edges, we increase the size of the matching by one edge. If we add s and t and direct the edges, the augmenting path becomes a directed path connecting s to t.

BERGE'S THEOREM. The matching M is maximum iff there is no augmenting path.

PROOF. Clearly, if there is an augmenting path then M is not maximum. Equivalently, if M is maximum then there is no augmenting path. Proving the other direction is more difficult. Suppose M is not maximum. Then there exists a matching N with $|N| > |M|$. We consider the symmetric difference obtained by removing the duplicate edges from their union,

$$M \oplus N \quad = \quad (M \cup N) - (M \cap N).$$

Since both sets are matchings, the edges of M are vertex-disjoint and so are the edges of N. It follows that each connected component of the graph $(V, M \oplus N)$ is either an alternating path or an alternating cycle. Every alternating cycle has the same number of edges from M and from N. Since N has more edges than M, it also contributes more edges to the symmetric difference. Hence, at least one component has more edges from N than from M. This is an augmenting path. ◻

Constructing a maximum matching. Berge's Theorem suggests we construct a maximum matching iteratively. Starting with the empty matching, $M = \emptyset$, we find an augmenting path and increase the size of the matching in each iteration until no further increase is possible. The number of iterations is less than the number of vertices. To explain how we find an augmenting path, we add vertices s and t to the graph, connecting s to all unmatched vertices in X and t to all unmatched vertices in Y. Furthermore, we direct all edges: from s to X, from X to Y if the edge is in $E - M$, from Y to X if the edge is in M, and from Y to t; see Figure 31. An augmenting path starts and ends with an edge in $E - M$. Prepending an edge from s and appending an edge to t, we get a directed path from s to t in the directed graph. Such a path can be found with breadth-first search, which works by storing active vertices in a queue and marking vertices that have already been visited. Initially, s is the only marked vertex and the only vertex in the queue. In each iteration, we remove the last vertex, x, from the end of the queue, mark all unmarked successors of x, and add these at the front to the queue. We halt the algorithm when t is added to the queue. If this never happens then there is no augmenting path and the matching is maximum. Otherwise, we extract a path from s to t by tracing it from the other direction, starting at t, adding one marked vertex at a time.

The breadth-first search algorithm takes constant time per edge. The number of edges is less than n^2, where $n = |V|$. It follows that an augmenting path can be found in time $O(n^2)$. The overall algorithms takes time $O(n^3)$ to construct a maximum matching.

Vertex covers. Running the algorithm to completion, we get a maximum matching, $M \subseteq E$. Let Y_0 contain all vertices in Y reachable from s and X_0 all vertices in X from which t is reachable; see Figure 32. No edge in M

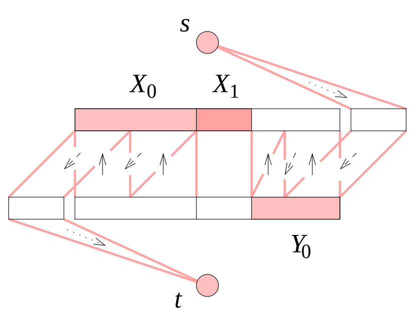

Figure 32: Schematic picture of the vertex set D consisting of the shaded portions of X and of Y. The vertices are ordered so that all edges in M are vertical.

connects a vertex in X_0 with a vertex in Y_0, else we would have an augmenting path. Furthermore, $|X_0 \cup Y_0| \leq |M|$ because each vertex in the union is incident to an edge in the matching. Letting X_1 contain the endpoints of the yet untouched edges in M, we set $D = X_0 \cup Y_0 \cup X_1$ and note that $|D| = |M|$. Furthermore, we observe that D *covers* all edges in E, that is, each edge has at least one endpoint in D.

We generalize this concept. Given a graph $G = (V, E)$, a *vertex cover* is a set $C \subseteq V$ such that each edge in E has at least one endpoint in C. It is *minimal* if it does not properly contain another vertex cover and *minimum* if there is no vertex cover with fewer vertices. Finding a minimal vertex cover is easy, eg. by greedily removing one vertex at a time, but finding a minimum vertex cover is a difficult computational problem for which no polynomial-time algorithm is known. However, if G is bipartite, we can use the maximum matching algorithm to construct a minimum vertex cover.

KÖNIG'S THEOREM. If $G = (V, E)$ is bipartite then the size of a minimum vertex cover is equal to the size of a maximum matching.

PROOF. Let X and Y be the parts of the graph, $C \subseteq V = X \dot{\cup} Y$ a minimum vertex cover, and $M \subseteq E$ a maximum matching. Then $|M| \leq |C|$ because C covers M. Since M is maximum, there is no augmenting path. It follows that the set $D \subseteq V$ (as defined above) covers all edges. Since C is minimum, we have $|C| \leq |D| = |M|$, which implies the claim. ∎

Neighborhood sizes. If the two parts of the bipartite graph have the same size it is sometimes possible to match every last vertex. We call a matching *perfect* if $|M| = |X| = |Y|$. There is an interesting relationship between the existence of a perfect matching and the number of neighbors a set of vertices has. Let $S \subseteq X$ and define its *neighborhood* as the set $N(S) \subseteq Y$ consisting of all vertices adjacent to at least one vertex in S.

HALL'S THEOREM. In a bipartite graph $G = (V, E)$ with equally large parts X and Y, there is a perfect matching iff $|N(S)| \geq |S|$ for every $S \subseteq X$.

PROOF. If all vertices of X can be matched then $|N(S)| \geq |S|$ simply because $N(S)$ contains all matched vertices in Y, and possibly more. The other direction is more difficult to prove. We show that $|N(S)| \geq |S|$ for all $S \subseteq X$ implies that X is a minimum vertex cover. By König's Theorem, there is a matching of the same size, and this matching necessarily connects to all vertices in X.

Let now $C \subseteq X \dot{\cup} Y$ be a minimum vertex cover and consider $S = X - C$. By definition of vertex cover, all neighbors of vertices in S are in $Y \cap C$. Hence, $|S| \leq |N(S)| \leq |Y \cap C|$. We therefore have

$$\begin{aligned} |C| &= |C \cap X| + |C \cap Y| \\ &\geq |C \cap X| + |S| \\ &= |C \cap X| + |X - C| \end{aligned}$$

which is equal to $|X|$. But X clearly covers all edges, which implies $|C| = |X|$. Hence, X is a minimum vertex cover, which implies the claim. ☐

Summary. Today, we have defined the marriage problem as constructing a maximum matching in a bipartite graph. We have seen that such a matching can be constructed in time cubic in the number of vertices. We have also seen connections between maximum matchings, minimum vertex covers, and sizes of neighborhoods.