Matchings and flows

ICTS-RRI Maths Circle, Bengaluru

12, 26 April 2025, 10:00 am to 1:00 pm

Matchings

Employees and tasks: On a certain morning, n employees show up at an office. The manager determines that there are m tasks that need to be completed that day. To complete a task an employee must be assigned to work on it for the entire day. Tasks need differing skills—not all employees possess the necessary training to perform all tasks. The manager knows the tasks that each employee can perform (see fig. 1). How can the manager determine if all m tasks can still be completed that day?

	t_1	t_2	t_3	t_4	t_5
w_1			+		
w_2	+	+			
w_3		+	+		
w_4		+	+		
w_5			+	+	+

Figure 1: Employees (w_1 to w_5) and tasks (t_1 to t_5)

Cabs and customers: At a certain time, a cab company has n taxis currently available; the company knows where each cab is located. The company also has requests from m customers, who would each like a cab. The company can determine (based on current traffic conditions) how long a particular cab will take to reach a particular customer. Can the cabs be assigned so that no customer needs to wait more than five minutes?

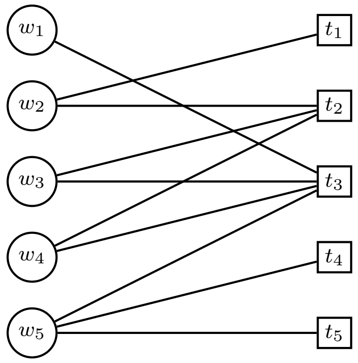

Figure 2: The graph for the office

Terminology

Problems such as these are called matching problems, or more precisely, *bipartite matching* problems, for there are two kinds of entities in each problem: employees-tasks and cabs-customers. Such problems are often visualised using *bipartite graphs*.

Bipartite graphs have two sets of vertices (nodes): A and B. In the first example, we take A to be the set of employees and B to be the set of tasks; in the second, we take A to be the set of cabs and B to be the set of customer requests. If vertex a in A is *compatible* with a vertex b in B then we place an edge (a, b) between them; e.g., we place an edge between employee w_4 and task t_3 because w_4 has been trained to for task t_3. The *input* to a matching problem is such a graph consisting of the vertex sets A and B, and the set of edges E: it is customary to write the three together as (A, B, E) and refer to this triple as the bipartite graph, and give it a name, such as G, G', H, etc.[1] There are other ways to represent graphs, of course. For example, we could draw a picture with the vertices in A on the left, and the vertices in B on the right, and connect a and b by a line whenever (a, b) is an edge of the graph. The lines need not be straight, the intermediate points hold no meaning, only the end points do, so we don't need to ensure that

$$\begin{aligned}
A &= \{w_1, w_2, w_3, w_4, w_5\};\\
B &= \{t_1, t_2, t_3, t_4, t_5\};\\
E &= \{(w_1, t_3), (w_2, t_1), (w_2, t_2), (w_3, t_2), (w_3, t_3),\\
&\quad (w_4, t_2), (w_4, t_3), (w_4, t_3), (w_4, t_4), (w_4, t_5)\}.
\end{aligned}$$

$$M_G = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Figure 3: The vertex set, the edge set, and the adjacency matrix

[1] The meaning of the term *graph* here is different from the more familiar notion associated with functions. In an abstract sense, the two are related, but that will not be important our discussion.

they do not cross; of course, as with every other diagram, keeping the drawings neat and symmetrical is often helpful. We will sometimes allow multiple edges (in parallel) between the same pair of vertices. Or one could represent the graph using a matrix, where the rows are for vertices in A and the columns are for vertices in B, and the entry in row a and column b is 1 if (a, b) is an edge, and 0 otherwise. This is the *adjacency matrix* of the graph—we write M_G for the adjacency matrix of G; note that M_G depends on the order we choose for the vertices of A and B. The entries of the adjacency matrices can be manipulated as numbers, by multiplying them, adding them, etc.; as a result what started off as combinatorial information takes on an algebraic life with many deep and beautiful consequences, which we will discuss some other time. Of these many ways of describing a graph, we will find the picture representation the most useful for our discussion. The *degree of a vertex a* is the number of edges that have a as an end-point; we write $\deg(a)$ to refer to the degree of a. In the graph of fig. 2, $\deg(w_2) = 2$.

A *matching* in a bipartite graph G is a set of disjoint edges. If (a, b) is an edge in the matching, then we say that a is matched to b. For example, in the graph of fig. 2 the set of edges

$$X = \{(w_4, t_2), (w_3, t_3), (w_5, t_4)\}.$$

is a matching, because no two edges in this set share a vertex. A *maximum cardinality matching* or just *maximum matching* is a matching with the most edges. A *perfect matching* is a matching where every vertex of the graph are matched.

Exploration I: The card puzzle

Take the standard deck of cards with four suits (spades, hearts, diamonds, clubs) and with 13 cards of each suit with ranks A, 2, ..., 9, 10, J, K, Q. Shuffle the deck if you wish. Now arrange the cards in 13 rows of 4 cards each. For example, consider the arrangement of cards shown in fig. 4.

I.1 Can you pick one card from each row so that you get all 13 ranks (they need not be of the same suit, of course)? (This might take some trial and error.) Try to model this as a matching problem.

I.2 What is the bipartite graph (A, B, E) corresponding to this problem? What do A and B represent? When would you connect a vertex a in A and a vertex b in B by an edge? Does your graph have parallel edges? What are the degrees of the vertices in A and B?

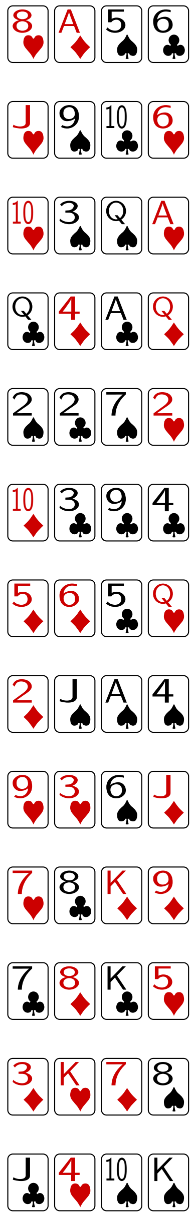
Figure 4: The card puzzle

I.3 Show that in every bipartite graph where all vertices have the same degree, say $r \geq 1$, both A and B must have the same number of vertices? Such graphs are said to be r-regular.

I.4 Make up some r-regular bipartite graphs ($r \geq 1$). Does each of them have a perfect matching?

Exploration II: Greedy

Is there a systematic way to find a maximum matching in a given bipartite graph? Let us try the greedy matching strategy shown in fig. 5.

II.1 Find a bipartite graph where this greedy method does not find the maximum matching.

II.2 How bad is the solution of the greedy method? Show that the greedy method always finds a matching that is at least half as large as the best, that is,

$$|\text{Greedy}(G)| \geq \frac{1}{2}|\text{Opt}(G)|.$$

Step 0: List the edges in some order: $e_1, e_2, \ldots, e_m$; start from the empty matching $X = \{\}$.

Step 1: Add e_1 to X.

Step 2: If e_2 is disjoint from e_1, add e_2 to X.

$\vdots \quad \vdots$

Step ℓ: If e_ℓ is disjoint from the edges already in X, add e_ℓ to X.

Step m: If e_m is disjoint from the edges already in X, add e_m to X.

Final step: Output X.

Figure 5: Greedy

Exploration III: Beyond greedy

Let us try another strategy. Recall that the greedy method maintains a matching X and tries to improve it (i.e., make it bigger) by adding another edge if it is not in conflict with an edge already in X. What else could one do? Look carefully at Greedy's behaviour in the example you constructed in Exploration II (i)? What did it miss?

Is *adding* an edge the only way to improve X? We are not able to add more edges because we are being obstructed by the edges already in X? Could we, perhaps, remove one edge from X, and create an opportunity for adding two? That too would result in a bigger matching. What else? After working out some examples, write down a new strategy; state it in a generally applicable form.

Figure 6: Greedier than greedy?

To help our thoughts, let us use the following terminology. Suppose we have a bipartite graph G and a matching X, which we want to improve.

Matched vertex, free vertex: A vertex v is a *matched vertex* (or saturated vertex) with respect to X if it is and end points of an edge in X; v is a *free vertex* (unmatched vertex) if it is not a matched vertex.

Matching edge: An edge (a, b) is a matching edge if (a, b) is in X.

X saturates S: We say that X saturates a set S of vertices if every vertex in S is matched with respect to X.

The greedy method when it makes progress ends up converting two free vertices into matched vertices. By looking at examples, see if you can discover a less short-sighted version of the greedy method, that succeeds where the greedy method fails.