Algorithmic Modeling and Complexity	**Fall 2003**

Lecture 8: 26 September

Lecturer: J. van Leeuwen *Scribes: Paulo Magalhães and Rui Coelho*

8.1 Overview

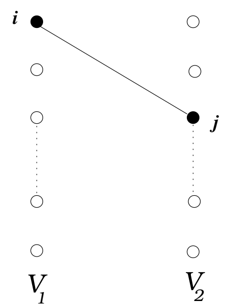
The assignment (or: *matching*) problem appears in the dilemma of assigning people to tasks, consultants to projects, interns to hospitals and so on, while respecting restrictions like availabilities, competences (abilities), and preferences. In this lecture we started with the general Bipartite Matching Problem, solving it by means of algorithms for network flow problems. We finished the lecture with the Stable Matching Problem in which people's *preferences* are taken into account in the assignment process.

8.2 Matching problems

We study the problem of assigning tasks by 'matching'. Define:

V_1 a finite set of persons,
V_2 a finite set of tasks.

If we assume that each person can do only *some* of the tasks, then the problem can be represented by a *bipartite* network[1] G as follows:



edge (i, j) means: person i can do task j.

We want to assign persons to tasks such that:

- every person is assigned to at most one task, and
- every task is assigned to at most one person,

[1] A network where the nodes are divided into two disjoint sets and there are only edges (i, j) with i, j belonging to different sets.

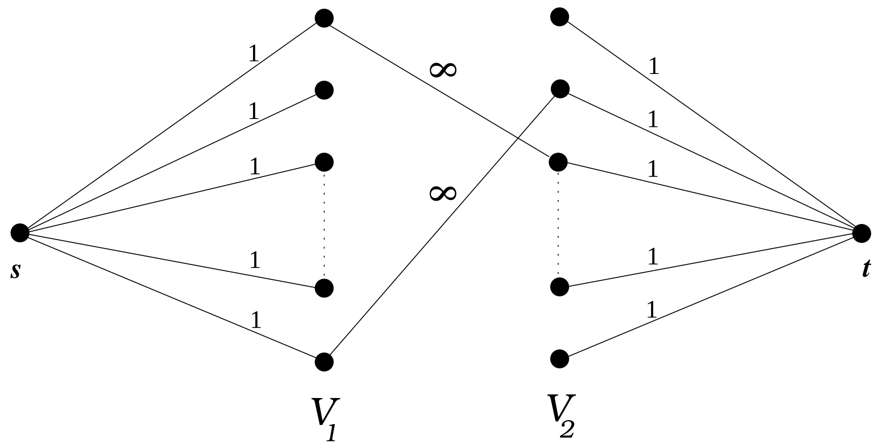
and some *cost criterion* is optimized. (If one allows that more than one task is assigned to a single person, then we speak of *scheduling* rather than matching.) In order to be precise we define a matching in a graph $G = (V, E)$, not necessarily bipartite as a set $M \subseteq E$ such that no vertex in V is incident to two different edges in M.

8.2.1 Maximum cardinality bipartite matching

In 'maximum cardinality bipartite matching' we want to match persons to tasks such that the total number of matched pairs is as large as possible. It will put the largest possible number of persons to work, and gets that largest number of tasks done in this shift. Thus we want to have a matching M of G such that M has maximum cardinality.

Theorem 8.1 *The maximum cardinality bipartite matching problem can be solved in polynomial time by reducing it to a maximum flow problem.*

Proof: Let $G = (V_1, V_2, E)$ be the instance of the maximum cardinality matching problem. Design a flow network G' as shown:



Step 1. Create source and target nodes s and t.

Step 2. Create edges from s to each node in V_1 and from each node in V_2 to t with capacity 1.

Step 3. Give the existing edges from nodes in V_1 to nodes in V_2 an unrestricted capacity (∞).

Claim 8.2 *Any maximum flow in G' corresponds to a maximum cardinality matching in G and vice-versa.*

To see this, first consider a maximum cardinality matching M in G and say it consists of k edges. Then there is a flow of value k in G', obtained by sending a flow of 1 through each of the k matched edges.

Conversely, suppose we have a maximum flow in G'.

Fact *If a flow network has integral capacities, then the value of the maximum flow is integral and there is a maximum flow with integral flows through the edges.*

Thus, take a maximum flow with integral flow values in G': it is seen that the flow values can only be 0 or 1. The edges that carry one unit of flow from V_1 to $_2$ form a matching of the size equal to the value of the maximum flow.

The two considerations together prove the claim and hence the theorem. ■

There are more direct ways of solving the maximum cardinality bipartite matching problem. Start with any (e.g. maximal) matching M in G. A node u is called *exposed* if it is *not* incident to a matched edge, i.e. u is a person or a task that is not assigned yet.

Definition 8.3 *An M-*augmenting path *is any* simple *path of odd length that starts at an exposed node and ends at an exposed node and consists alternately of non-matched and matched edges.*

Note that an M-augmenting path necessarily ends with a non-matched edge as well. If there exists an M-augmenting path, then exchanging the matched edges by the non-matched edges along the path in the matching results in a matching M' that is bigger by 1 matched pair! For general, i.e. even non-bipartite networks the following result is known.

Theorem 8.4 (Berge, 1957) *A matching M has maximum cardinality if and only if G contains no M-augmenting path.*

The maximum cardinality bipartite matching problem can be 'easily' solved by finding an M-augmenting path and augmenting an initial matching iteratively until no further M-augmenting paths exist.

Theorem 8.5 (Hopcroft and Karp, 1973; Even and Tarjan, 1975) *The maximum cardinality bipartite matching can be solved in $O(\sqrt{n}.m)$ time, with n the number of nodes and m the number of edges.*

We note that the given time-bound holds more generally for the best case of maximum flow algorithms but not necessarily for the most practical ones in general.

8.2.2 Characterisations

The question arises whether we can say anything about the possibility in a given matching problem of matching e.g. all persons ('all persons have work to do') or of matching all persons *and* all tasks ('perfect matching').

The questions arises whether each instance of the matching problem allows for a matching in which all persons and all tasks are matched (a so-called perfect matching).

Theorem 8.6 (König 1931) *In a bipartite network the size of a maximum number of matching is equal to the size of a minimum vertex cover.*

Proof:

We refer again to the proof of Theorem 8.1 where we showed: the size of a maximum number of matching is equal to the value of the maximum flow in G'. We now show another property of the flow network G'.

Claim 8.7 *There is a vertex cover of size k in G if and only if there is an* s-t cut[2] *of capacity k in G'.*

To show this, let C be a vertex cover of size k in G. Consider the set of edges C' consisting of: edges (s, i) with $i \in V_1 \cap C$, and edges (i, t) with $i \in V_2 \cap C$. C' is a *s-t cut* because any path from s to t must pass through a node of the vertex cover. Clearly C' has capacity k.

Conversely, say C is an *s-t cut* of capacity k. Then C consists entirely of edges from s to V_1, and from V_2 to t (as the other edges have capacity ∞). The nodes in V_1 and in V_2 incident to these edges form a vertex cover of exactly size k.

By the claim, the minimum size of a vertex cover in G is equal to the value of the minimum capacity *s-t cut* in G'.

Fact *In flow networks, the value of the maximum flow is equal to the minimum capacity of any s-t cut.*

This proves the theorem. ■

Exercise Let M be a maxim*al* matching, OPT the size of the maximum matching in a given bipartite network. Show that $|M| \geq \frac{1}{2}OPT$. (*Hint.* Argue that the endpoints of the edges in M form a vertex cover. Then use König's theorem.) Show that this holds for arbitrary networks as well.

König's theorem is very helpful for answering our questions. First, consider the problem of determining whether a matching can be found that matches all persons: a V_1*-perfect matching.*

Definition 8.8 *For any set $S \subseteq V_1$, let $NB(S) = \{j|$ there is an i in S such that $(i, j) \in E\}$ (the set of distinct nodes to which the nodes in S are connected).*

Obviously $NB(S) \subseteq V_2$ for every $S \subseteq V_1$.

Theorem 8.9 (Hall, 1935) *A bipartite network has a V_1-perfect matching if and only if for every subset $S \subseteq V_1$ one has $|NB(S)| \geq |S|$.*

Proof:

[2]An *s-t cut* is a set of edges whose removal disconnects s and t. The capacity of an *s-t cut* is the sum of the capacities of the edges in the cut.

($\Rightarrow$) Trivial.

($\Leftarrow$) By König's theorem the result follows once we show that under the given condition, every vertex cover of G must have size $\geq |V_1|$.

Let T be any vertex cover of G. Let S be the set of nodes in V_1 that do *not* belong to T (thus the remaining $|V_1| - |S|$ nodes of V_1 do belong to T). In order to be a vertex cover, the edges incident to S must all have their other endpoint in T: there are exactly $|NB(S)|$ of these nodes. Hence

$$|T| = (|V_1| - |S|) + |NB(S)| = |V_1| + (|NB(S)| - |S|) \geq |V_1|$$

This proves the result. ∎

The given result is also known as Hall's *marriage theorem*. Let V_1 be a set of boys and V_2 a set of girls. Let an edge (i, j) represent whether boy i knows girl j. Then Hall's theorem says that the boys can all be matched to a girl they know if and only if any subset of the boys collectively knows at least as many girls.

Exercise. Formulate a version of Hall's theorem in terms of V_2. Phrase the corresponding condition in terms of sets of tasks and the sets of persons that can carry them out.

8.2.3 Perfect matchings

Let a *perfect matching* be a matching in which all nodes are matched (so necessarily $|V_1| = |V_2|$ in this case). We now formulate some classical characterisations for the existence of perfect bipartite matchings.

Lemma 8.10 (Frobenius 1917) *In a bipartite networks a perfect matching exists if and only if every vertex cover has size greater or equal than* $\frac{1}{2}(|V_1| + |V_2|)$.

Proof: ($\Rightarrow$) In a perfect matching in a bipartite network all nodes in V_1 are matched with all nodes in V_2. A vertex cover must contain at least one endpoint of each edge in the matching. Thus the vertex cover has size $\geq |V_1| = |V_2| = \frac{1}{2}(|V_1| + |V_2|)$.

($\Leftarrow$) Observe that necessarily $|V_1| = |V_2|$: observe that both V_1 and V_2 are vertex covers of G and if they both are to have a size $\geq \frac{1}{2}(|V_1| + |V_2|)$ then they must have *equal* size. As the minimum size of a vertex cover is then $|V_1| = |V_2|$, it follows by König's theorem that the size of the maximum matching is equal to $|V_1| = |V_2|$. Therefore we have a perfect matching. ∎

Let's now consider the case of maximum cardinality matching in *regular* bipartite networks: a bipartite network in which all nodes equal degree.

Theorem 8.11 (König 1916) *Every regular bipartite network has a perfect matching.*

Proof: A bipartite network that is regular necessarily has: $|V_1| = |V_2|$. Say the nodes have degree d. It follows that G has precisely $d \cdot |V_1|$ edges.

Let T be any vertex cover. Any single node $\in T$ can cover d edges. In order to cover all edges, it follows that T must contain at least $|V_1|$ nodes. Now use Frobenius' lemma. ∎

One may hope that finding perfect matchings, if they exist, is easier than running a general maximum cardinality bipartite matching solver. This seems indeed to be the case.

Theorem 8.12 (Cole, Ost, and Schirra, 2001) *Concrete perfect matchings in regular bipartite graphs can be determined in linear time.*

Finally, consider the following Shared Tasks Problem. Suppose an edge (i, j) means that person i must do task j *sometime* (e.g. assemble part of object j). Thus every person must do all tasks to which he is connected by an edge, one task per person at a time. Let a *shift* be any matching of persons to tasks. Our goal is to schedule the assignments in the smallest number of *shifts*. Let d_{max} be the maximum degree of any node in G.

Theorem 8.13 *The Shared Tasks Problem can be solved optimally in d_{max} shifts.*

Proof: Clearly d_{max} shifts are required because the person or task with degree d_{max} must be assigned in at least this number of distinct shifts.

Conversely, consider any bipartite network G with maximum degree d_{max}. Refer to all nodes and edges of G as 'black'. Assume w.l.o.g. that $|V_1| \geq |V_2|$ and suppose k edges emanate from V_1. Add $|V_1| - |V_2|$ red tasks to G. Now complete G to a regular bipartite network of degree d_{max}: simply add $d_{max} \cdot |V_1| - k$ red edges to the nodes of V_1 that don't have full degree yet and connect them to any of the black or red tasks that don't have full degree yet either. Verify that this indeed gives a regular network.

By Theorem 8.11, G contains a perfect matching. In fact, it follows that G 'decomposes' into precisely d_{max} perfect matchings. Now implement each perfect matching as a shift: assign persons and tasks according to the black edges in the perfect matching, ignore the red edges. This gives a schedule of precisely d_{max} shifts. ∎

8.3 Stable matchings

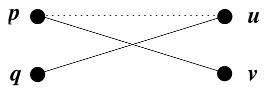
We will now study an especially interesting case of the perfect matching problem. Let's assume again that $|V_1| = |V_2| = n$.

Consider the case where all persons can do every task but where the persons and the tasks have clear preferences. More specifically, assume that:

- every person has an ordered list of all the tasks, ordered by decreasing preference, and
- every task has also an ordered list of persons that can be assign to it, ordered by decreasing preference.

The question is whether a (perfect) matching can be determined in which both the persons and the tasks are assigned in a way that such that 'optimum satisfaction' is reached. We will show that a 'peaceful equilibrium' exists!

Definition 8.14 *A matching is called* unstable *if there is a person p and a task u such that: p and u are not matched to each other but with v being the match of p and q the match of u, the following is true: p prefers u over v, and u prefers p over q. Pictorially:*



Definition 8.15 *A matching is stable if it is not unstable.*

Theorem 8.16 (Gale and Shapley, 1962) *Stable matchings exist and can be found in easy polynomial time.*

The algorithm that proves it interpretes the persons as 'men' and the tasks as 'women'. In this form, the problem is known as *the stable marriage problem.* The algorithm works through a series of proposals where every time a 'free' man proposes to the most preferred woman on his list that did not rejected him previously. If a man and a woman are temporarily matched, they are called 'engaged'. If a woman is proposed to by a man whom she prefers over her current fiancé, she 'frees' her current fiancé and engages with the more preferred man.

Algorithm GS

let every person be free

while there are free men left

begin

    $m :=$ any free man

    $w :=$ first woman on m's preference list to whom m did not propose before

    if w is free **then**

        m and w become engaged

    else

        if w prefers m to her current fiance m' **then**

            m' becomes free

            m and w become engaged

        else

            w rejects m (and m remains free)

end

return the engaged pairs

We now show that this greedy algorithm solves the problem.

Lemma 8.17 *Algorithm GS terminates, and terminates with a perfect matching.*

Proof: Observe that once a woman becomes engaged, she will be engaged forever. Also, a man proposes to the women on his list one after the other (every time he is free), in decreasing order. This guarantees termination after at most n^2 proposals.

Suppose there is man m who has proposed to all women on his list and has not become engaged. Consider what happened when m proposed to the last woman w on his list. Because m was down to the last woman on his list, he must have proposed and (eventually) been rejected by all $n-1$ women higher on his list at some earlier time. As they all rejected him, these women must all be engaged, to $n-1$ men. Thus w cannot be engaged at the moment m proposes to her, as there is no man other than m left for her. Contradiction.

Thus upon termination, GS returns a perfect matching. ∎

Exercise. Show that by choosing suitable datastructures, algorithm GS can be implemented to run in $O(n^2 \log n)$ time.

Lemma 8.18 *Algorithm GS returns a stable matching.*

Proof: Suppose that the resulting matching is not stable. Then, by definition, there must be a man p and a woman u such that p prefers u but is engaged to v, while woman u prefers p but is engaged to q. ('Preference' here means 'occurs higher on the preference list'.)

Note that p must have proposed to u earlier. If u was free when p proposed to her, then u would not have swapped p for q (because p is higher on her list). This means that when p proposed to u, u rejected p because she was already engaged to some other man that she preferred over p. Then the only way of u getting matched with q is if she preferred him to the previous man. Then q must be preferred over p also, contradicting our assumption.

Thus the matching must be stable. ∎

Algorithm GS allows an *arbitrary choice* of a free man at the beginning of every iteration (m := any free man). Despite this, one can show the following somewhat surprising fact.

Theorem 8.19 *Every run of algorithm GS results in the* same *stable matching.*

The proof is based on the fact that one can show that in algorithm GS, every man gets the *best* partner he can get in *any* stable matching. In contrast, every woman gets the *worst* partner she can get in any stable matching.

The idea of the stable matching algorithm admits many variations [4].

References

[1] R. Cole, K. Ost, S. Schirra. Edge-coloring bipartite multigraphs in $O(E \log D)$ time. *Combinatorica* 21 (2001) 5-12.

[2] S. Even, R.E. Tarjan. Network flow and testing graph connectivity. *SIAM J. on Computing* 4 (1975) 507-518.

[3] D. Gale, L.S. Shapley. College admissions and the stability of marriage. em American Math. Monthly 69 (1962) 9-15.

[4] D. Gusfield, R.W. Irving. *The stable marriage problem: Structure and algorithms.* The MIT Press, Cambridge, MA, 1989.

[5] J. Hopcroft, R.M. Karp. An $n^{\frac{5}{2}}$ algorithm for maximum matching in bipartite graphs. *SIAM J. on Computing* 2 (1973) 225-231.

[6] D. König. "Graphen und Matrizen". *Matematikai és Fizikai Lapok* 38 (1931) 116-119.