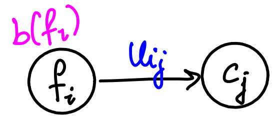
MATH 566: Lecture 3 (08/27/2024)

Today: * seat sharing problem
* assignment problem
* definitions

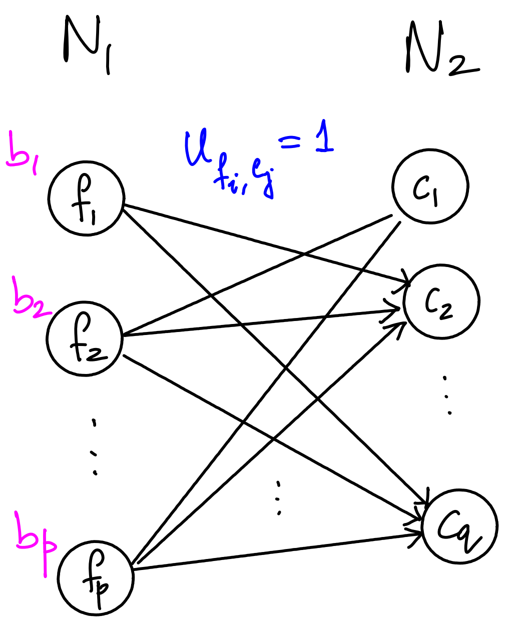
Seat Sharing Problem

(AMO 1.8, page 21) Several families are planning a shared car trip on scenic drives in the Cascades in Washington. To minimize the possibility of any quarrels, the organizers of the trip want to assign individuals to cars so that **no two members of a family are in the same car**. Formulate this problem as a network flow problem.

Let there p families, with b_i members for $1 \le i \le p$. And let there be q cars, with u_j seats, for $1 \le j \le q$. We start with a node for each family and a node for each car. Let $N_1 = \{f_1, \dots, f_p\}$ be the set of family nodes and $N_2 = \{c_1, \dots, c_q\}$ be the car nodes.



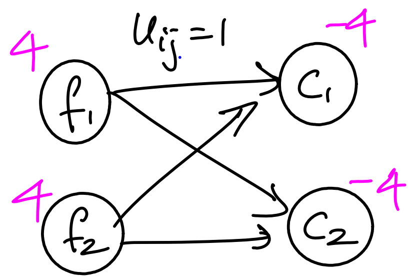
We set $b(f_i) = b_i$, i.e., the supply of family node f_i as b_i.



To capture the restriction that no two members of a family can be seated in one car, we add arcs from each f_i to each c_j, and set their capacities to 1.

Notice that we add an arc from each f_i to each c_j — as a member from any family could be sent to any car. We add a total of pq arcs, each with $u_{(f_i,c_j)} = 1$.

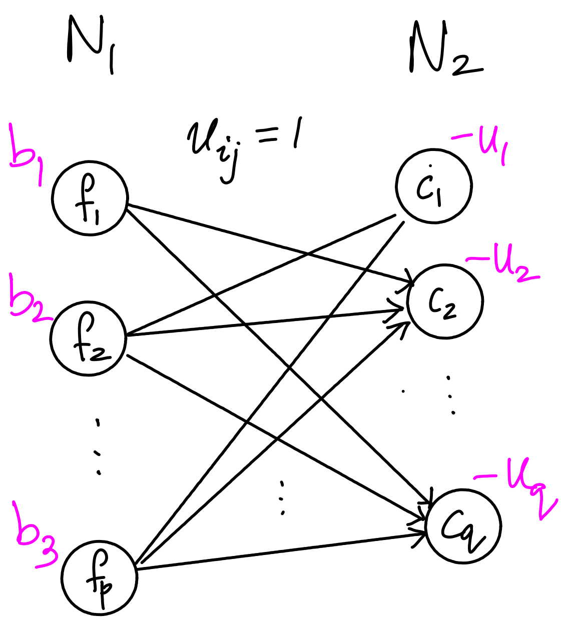
Assumption We assume there exists a feasible assignment of people to cars. There could be trivial instances where we cannot seat all members without avoiding clashes, e.g., see the network below:



Let the capacities of cars c_1 and c_2 be 4 each. We can set them as demands, but there is no feasible arrangement that avoids in-family clashes!

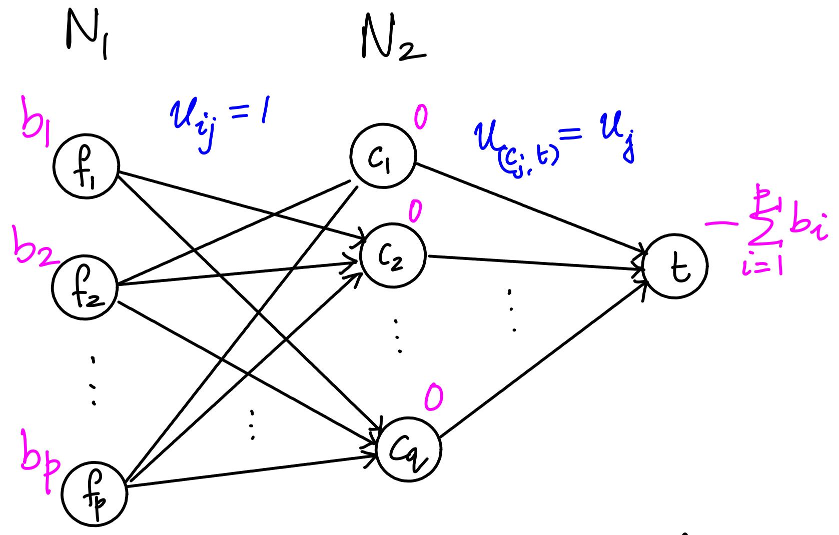
In fact, finding if a given network has a feasible flow can itself be modeled as another network flow problem! More on this topic in a bit...

Back to the problem now.



If we assume further that $\sum_i b_i = \sum_j u_j$, then we could set $b(c_j) = -u_j$, and we have a valid MCF problem (as a transportation feasibility problem, since no c_{ij}'s are used).

But more generally, there could be more total seats than there are people to be seated, i.e., $\sum u_j > \sum b_i$. Hence we consider the following more general model.

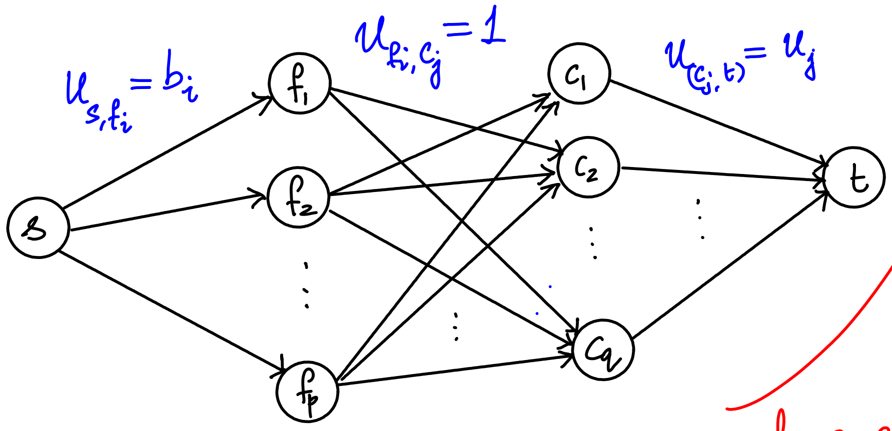


We add a terminus node t, and arcs (c_j,t) with $u_{(c_j,t)}=u_j$, the # seats available in car c_j. We set $b(j)=0$ for all $j\in N_2$, i.e., car nodes are transshipment nodes. And we set $b(t)=-\sum_i b_i$.

By setting $b(t)=-\sum_{i\in N_1} b(i)$ $\left(\text{i.e., } -\sum_{\text{families}}(\#\text{ members})\right)$, we ensure that all members from each family is assigned to some car.

This more general model allows one to accommodate some other variations/generalizations — see the problem in Homework 1.

The original problem could be modeled also as a max flow problem:



If the value of the max flow is $\sum_{f_i\in N_1} b_i$, then there is a feasible seating arrangement, and that is ~~specified~~ specified by the x_{f_i,c_j} values that are 1.

so, each arc (s,f_i) is saturated

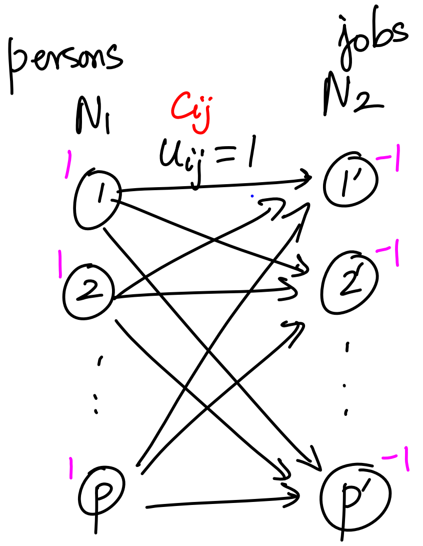
We saw shortest path (SP), max-flow (MF), min-cost flow (MCF), circulation, and transportation problems. Here is one more class of problems.

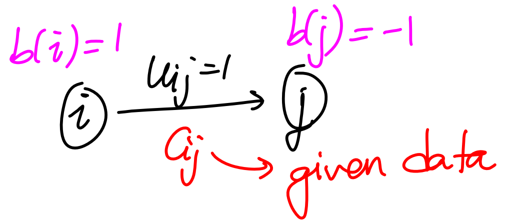
6. Assignment problem

Given p persons and p jobs, the goal is to assign each person to one job, such that the overall cost of the assignments is minimized. You are also given costs c_{ij} for assigning person i to job j.

We can model this problem as a case of MCF with node set $N = N_1 \cup N_2$. The network has a bipartite structure, similar to the case of the transportation problem.

$(i,j') \in A$ has $i \in N_1$ and $j' \in N_2$





We need not specify $u_{ij'} = 1$ here! Could leave them as $u_{ij'} = +\infty$, the default values, since $b(i) = 1 \; \forall i \in N_1$ and $b(j') = -1 \; \forall j' \in N_2$.

When formulating a network flow problem, you must describe the structure of the network, i.e., specify the node and arc sets N and A, and specify all associated parameters — $b(i) \; \forall i \in N$, $c_{ij}, l_{ij}, u_{ij} \; \forall (i,j) \in A$.

Default values: $b(i) = 0$, $c_{ij} = 0$, $l_{ij} = 0$, $u_{ij} = +\infty$.
If not specified, a parameter is assumed to be at its default value.

Definitions for Networks

We have seen some of them informally, today we will list them formally.

$G=(N,A)$ is the **network** with node set N and arc set A.

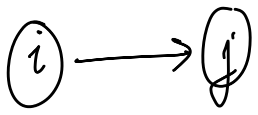
→ "network" and "graph" used interchangeably

$|N| = n$ (# nodes) $\quad N = \{1,2,\dots,n\}$

$|A| = m$ (# arcs) $\quad A = \{(1,2),(1,3),\dots,(i,j),\dots\}$

Directed network: edges or arcs of the network are directed

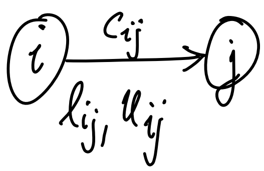
(i,j)



$(i,j) \neq (j,i)$ in a directed network.

We assume the network is directed by default.

node i : **tail**
node j : **head**



(i,j) is an $\overbrace{\text{outgoing}}^{\text{also, outarc}}$ arc of node i, and an $\overbrace{\text{incoming}}^{\text{also, inarc}}$ arc of node j.

If $(i,j) \in A$, then j is **adjacent** to i. Notice i is **not** adjacent to j because of (i,j).

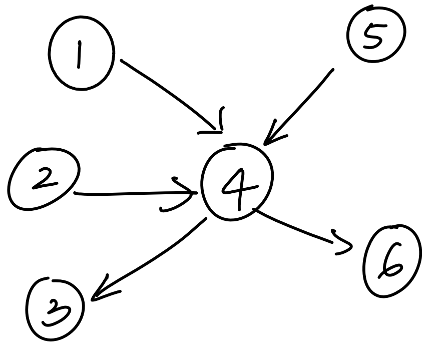
Degree

indegree(i) = # arcs incoming to node i

outdegree (i) = # arcs outgoing from node i

degree (i) = indegree(i) + outdegree(i).

Example



indegree$(4) = 3$

outdegree $(4) = 2$

degree $(4) = 5$

Adjacency list

→ perhaps the first data structure for networks!

Arc adjacency list $A(i) = \{(i,j) \in A, j \in N\}$, e.g. $A(4) = \{(4,3), (4,6)\}$.

Node adjacency list $A(i) = \{j \in N : (i,j) \in A\}$, e.g., $A(4) = \{3, 6\}$.

Notice that $|A(i)| =$ outdegree of i, and

$$\sum_{i \in N} |A(i)| = m \quad (\text{total \# arcs}).$$

With arcs numbered $1, 2, \ldots, m$, it is more efficient to store the corresponding arc numbers (or indices), instead of pairs (i,j).

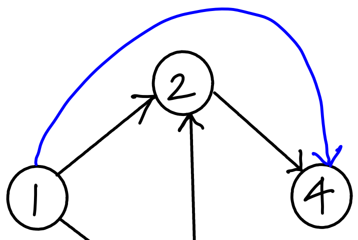
Note that $A(i)$ defines adjacency based on ~~out~~arcs. We can equivalently define inarc lists $AI(i)$. In practice, it often helps to initialize and use both sets of lists.

Subgraph A graph $G' = (N', A')$ with $N' \subseteq N$ and $A' \subseteq A$ is a **subgraph** of G.

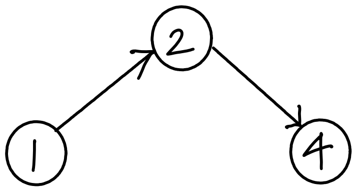
An **induced subgraph** is a subgraph that contains each arc of A with nodes in N', i.e., each such arc is present in A'.

A **spanning subgraph** has $N' = N$.

Example

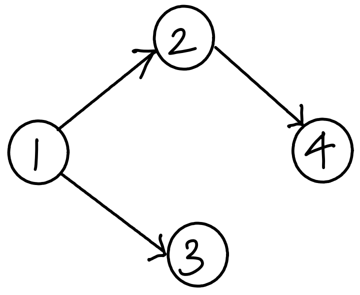


G



G' is an induced subgraph of G.

But is not induced if (1,4) were present



G'' is a spanning subgraph