CS612

Algorithms for Electronic Design Automation

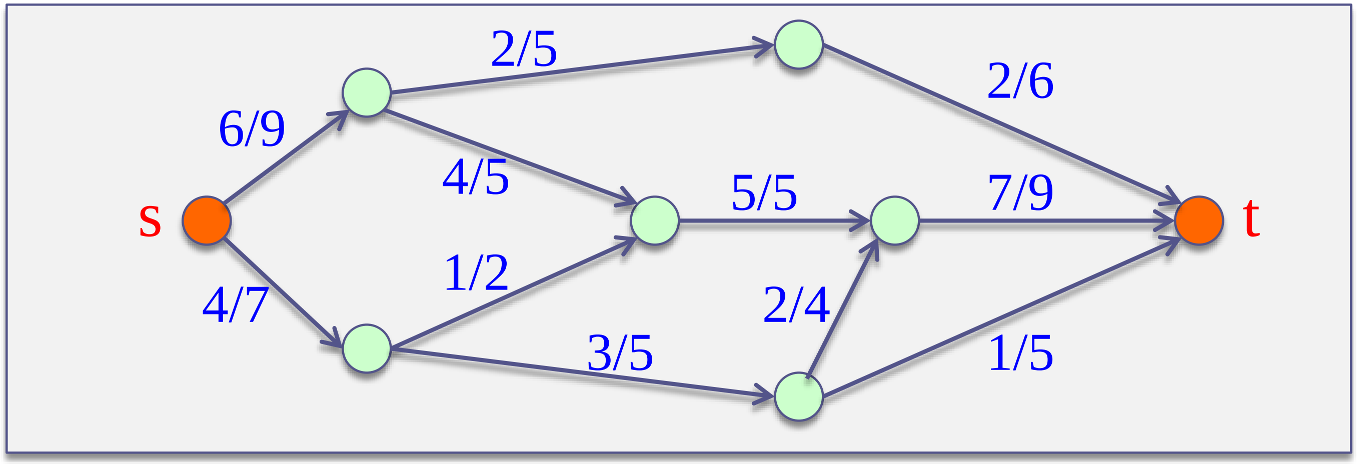
Lecture 8

Network Flow Based Modeling

Mustafa Ozdal

Flow Network Definition

- Given a directed graph $G = (V, E)$:
  - Each edge (u, v) has capacity $c(u,v) \geq 0$
  - Each edge (u, v) has flow $f(u, v) \geq 0$
  - A special source vertex s
  - A special sink vertex t



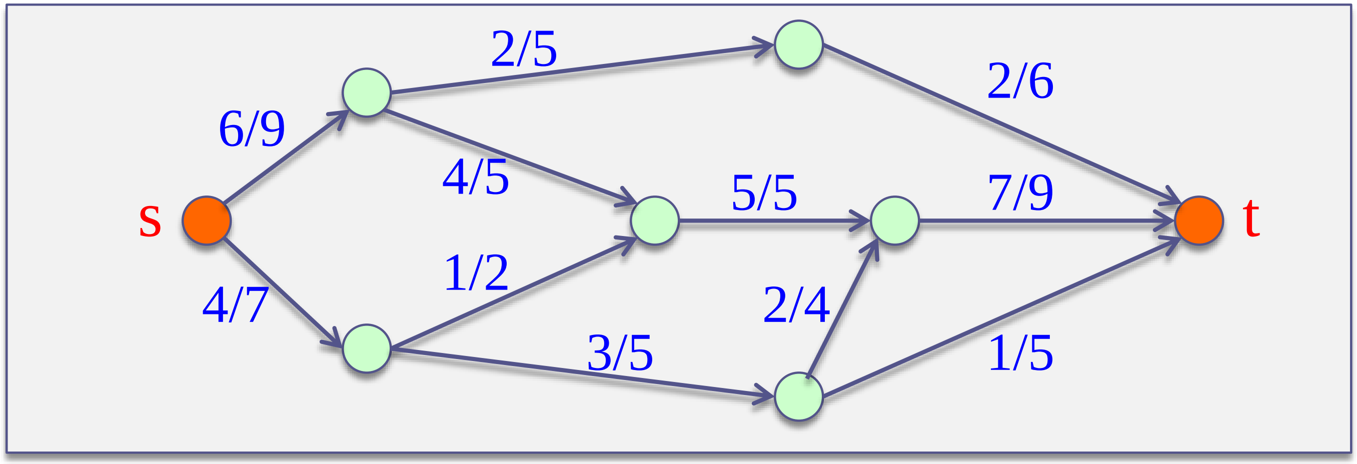
Flow f and capacity c values for each edge shown as f/c

Flow Constraints

- Capacity constraints: $0 \le f(u, v) \le c(u, v)$ for each edge (u, v)
- Flow conservation: For all $u \in V - \{s, t\}$, we must have:

$$\sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$$

Total incoming flow to u = Total outgoing flow from u



Flow f and capacity c values for each edge shown as f/c

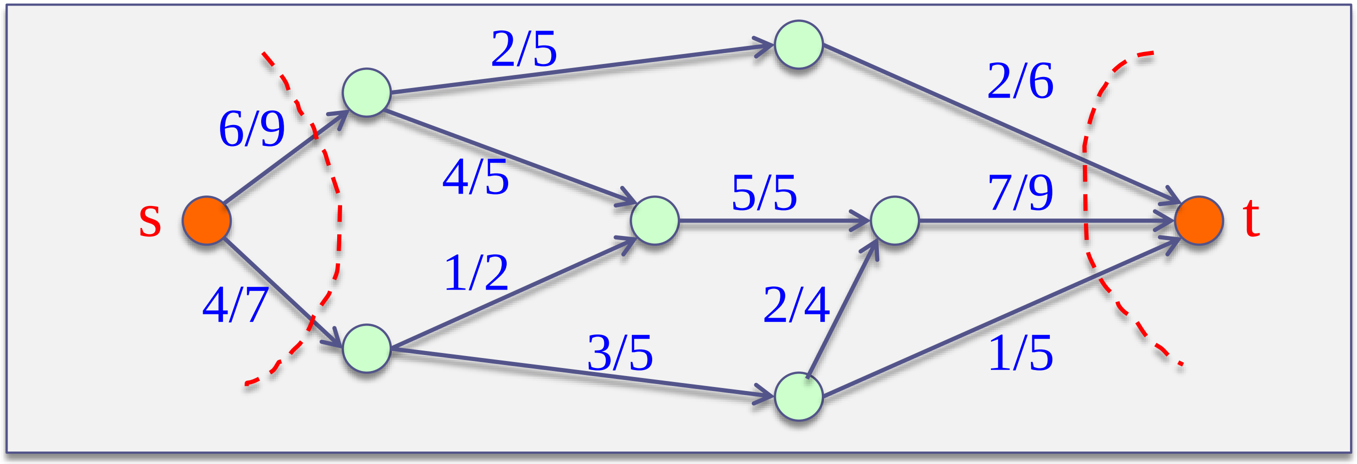
Network Flow

- The total flow through the network is defined as:

  the net flow out of source vertex s

  or equivalently:

  the net flow to the sink vertex t



Total flow = 10

Max Flow Problem

- Given a flow network, determine the flow values through each edge such that:
  - The capacity constraints are satisfied
  - The flow conservation constraints are satisfied
  - The total flow value is maximized

- *Integrality theorem*: If all edge capacities are integers, then it is guaranteed that there exists an optimal solution with integer flow values.

Max Flow Problem

- Max flow problem is polynomial-time solvable.
- In practice, we can model it as a linear programming (LP) problem, and make use of efficient linear solvers.
- If all edge capacities are integers, it is guaranteed that the corresponding LP model is unimodular

  $\Longrightarrow$ Linear solver will return a solution with integer values
- In practice, can handle reasonably large problems.

Bipartite Matching Problem

Many practical problems can be modeled as max flow problems.

Exercise:

There are n students who want to do internship, and there are m companies. Each student marks 3 companies as his/her preference. Your task is to assign the students to companies such that:

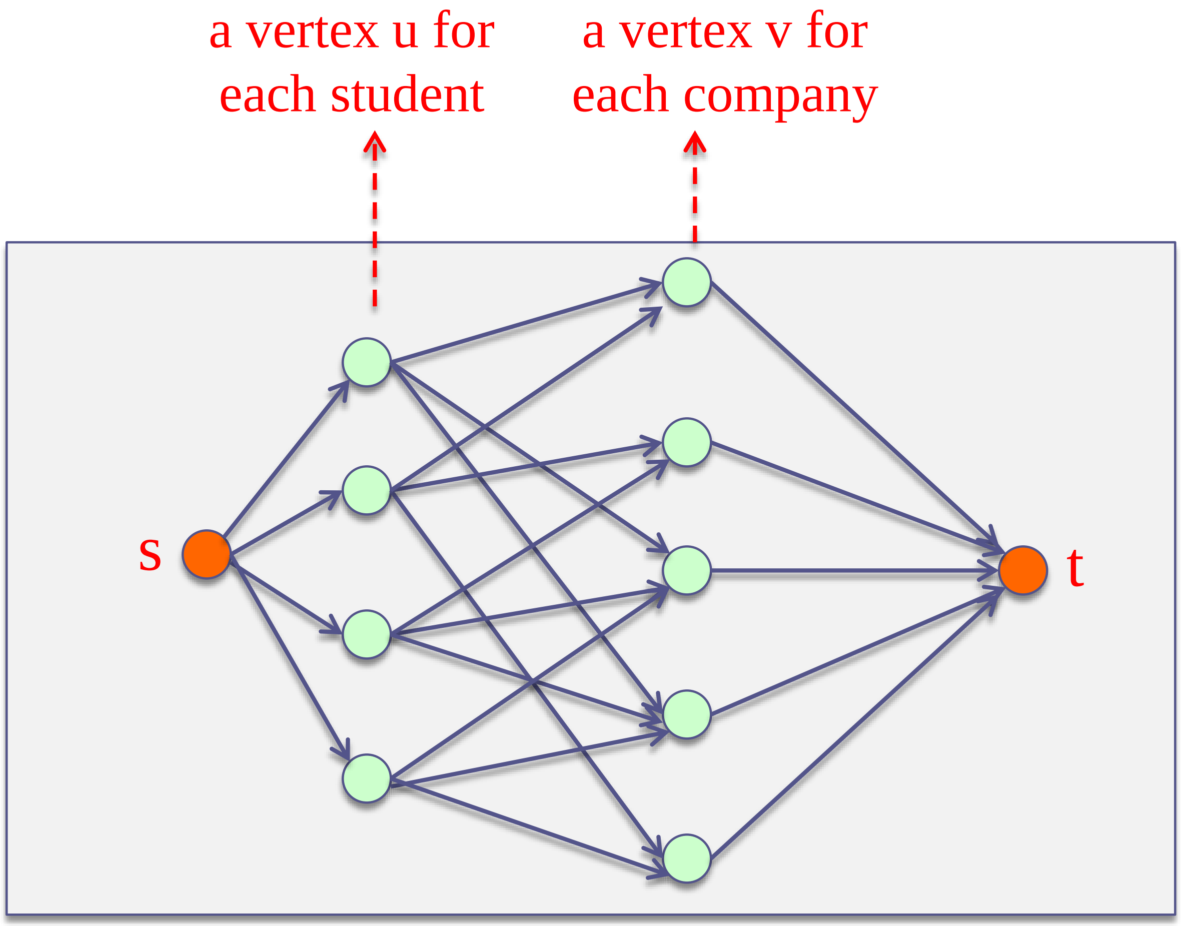
Each student is assigned to 1 company, and vice versa.

A student is not assigned to a company (s)he doesn't prefer.

The number of students assigned is maximized.

Use network flow to model your algorithm.

Solution



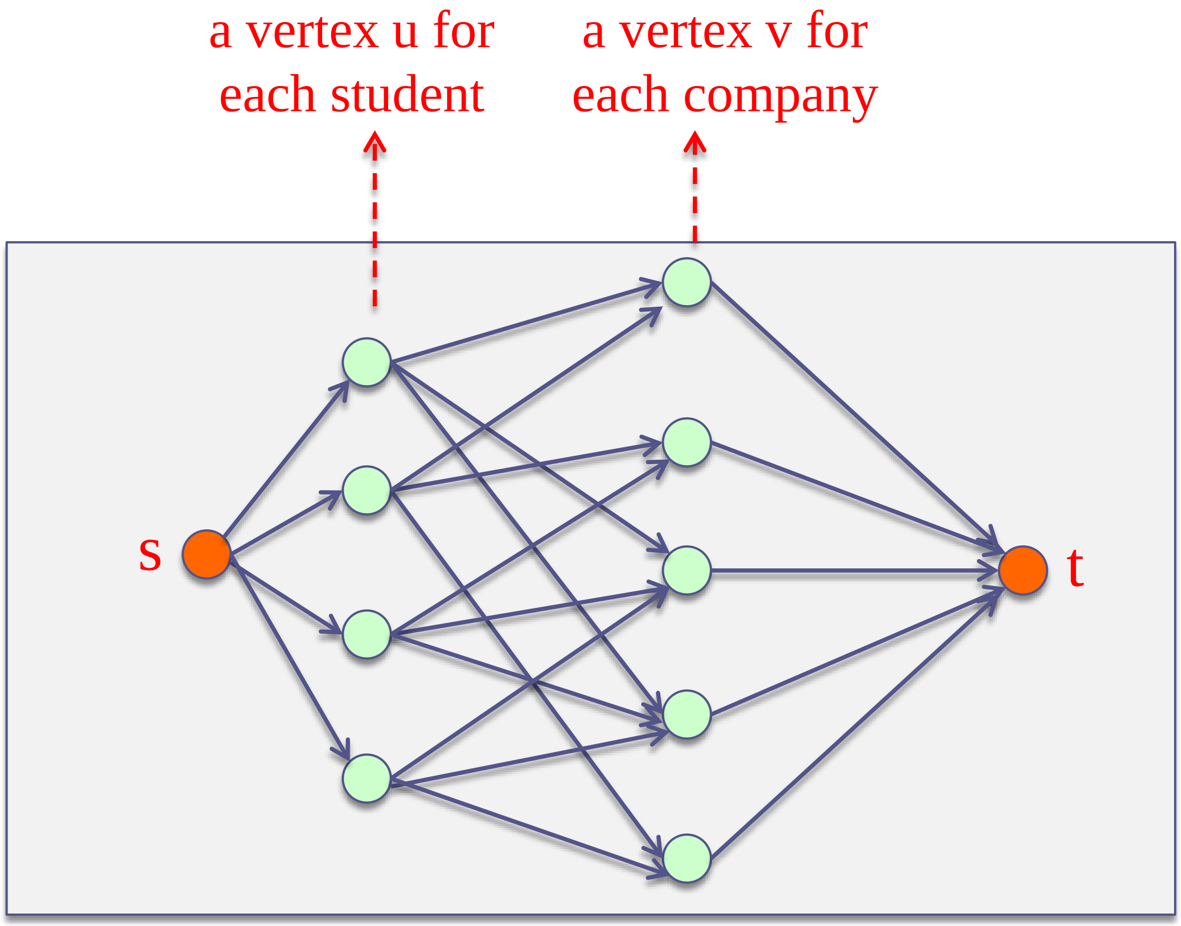
An edge from source s to each student vertex u.

An edge from company vertex v to sink t.

Create edge (u, v) iff student u prefers company v.

All edge capacities are 1.

Solution



Compute the max flow from source s to sink t.

Total flow = # of assignments

If edge (u,v) has non-zero flow, assign student u to company v.

Optimality Proof

1. Any student assignment with size $|A|$ can be mapped to a flow solution with size $|A|$.
2. Any flow solution with size $|F|$ can be mapped to a student assignment with size $|F|$.
3. The max-flow algorithm returns the solution with max total flow $|F_{max}|$. This solution can be mapped to a student assignment with the same size due to (2).
4. If there was a better student assignment with size $|A_{max}|$, where $|A_{max}| > |F_{max}|$, we would be able to map it to a flow solution with size $|A_{max}|$ due to (1). But, this would be a contradiction because $|F_{max}|$ is the maximum flow achievable.

Hence, the assignment obtained by mapping the max-flow solution must be optimal.

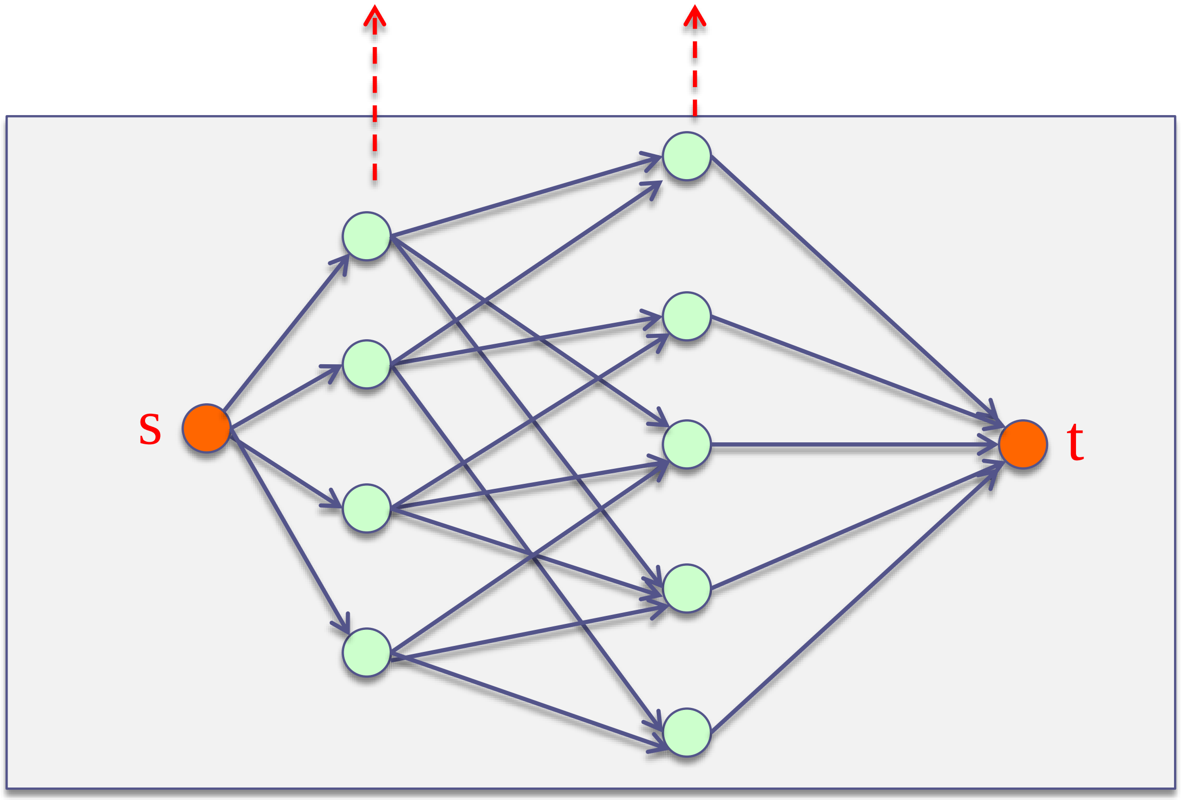
Exercise

There are n students and m courses. Each student indicates preference for 8 courses. You are supposed to assign courses to all students such that:

- A student is not assigned more than 5 courses.
- A course does not contain more than 20 students.
- A student is not assigned a course that (s)he doesn't prefer.
- The number of courses assigned to all students is maximized.

Use network flow to model your algorithm.

Solution

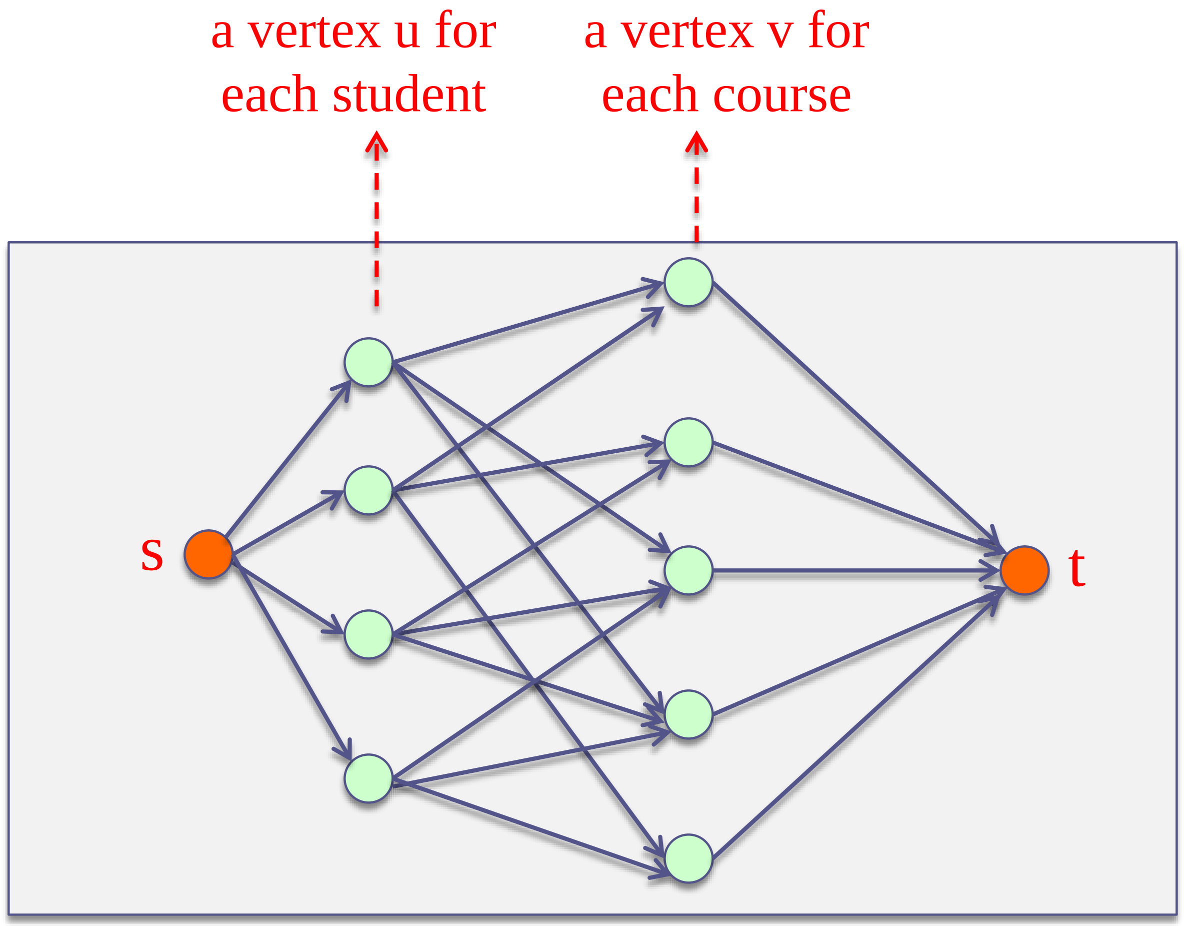


An edge from source s to each student vertex u with capacity 5.

An edge from course vertex v to sink t with capacity 20.

Create edge (u, v) with capacity 1 iff student u prefers course v.

Solution



Compute the max flow from source s to sink t.

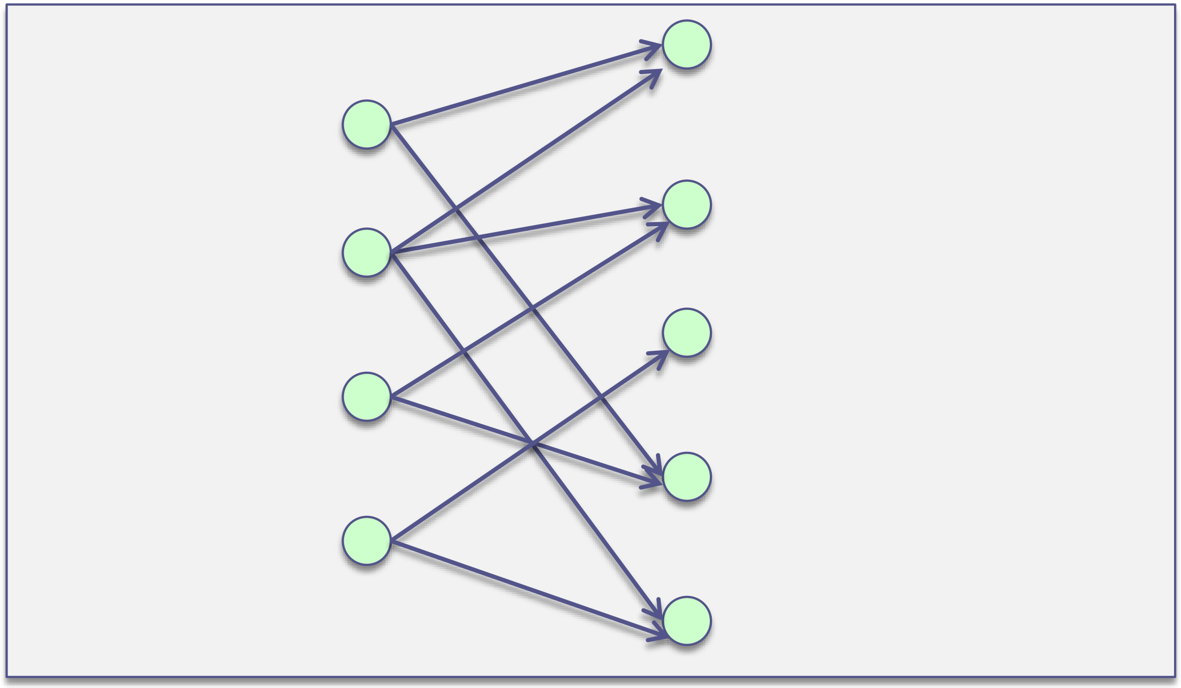
Total flow = # of assignments

If edge (u,v) has non-zero flow, assign student u to course v.

Proof: (1) Assignment → Flow

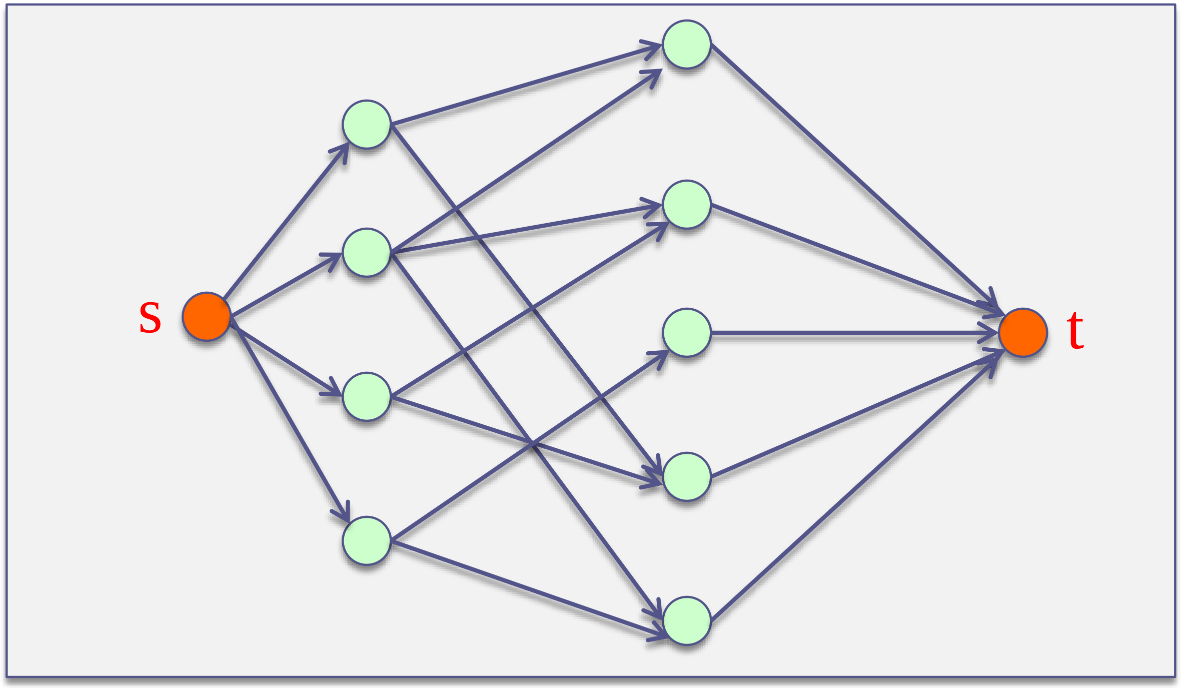
(1) <u>Show that any assignment can be mapped to a valid flow with the same size</u>

Given an assignment of size |S|:



Proof: (1) Assignment → Flow

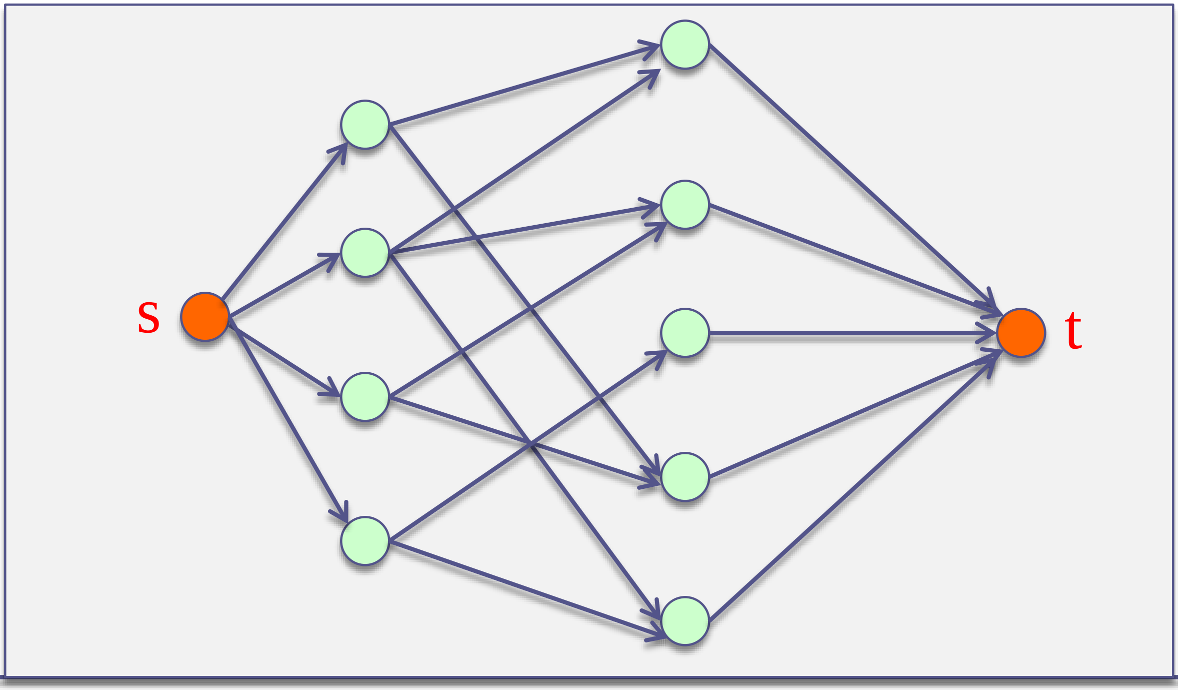
Create a network as suggested in the solution. For flow to be valid, we need to satisfy 2 conditions: 1) flow conservation, 2) capacity constraints



Proof: (1) Assignment → Flow

Flow conservation:

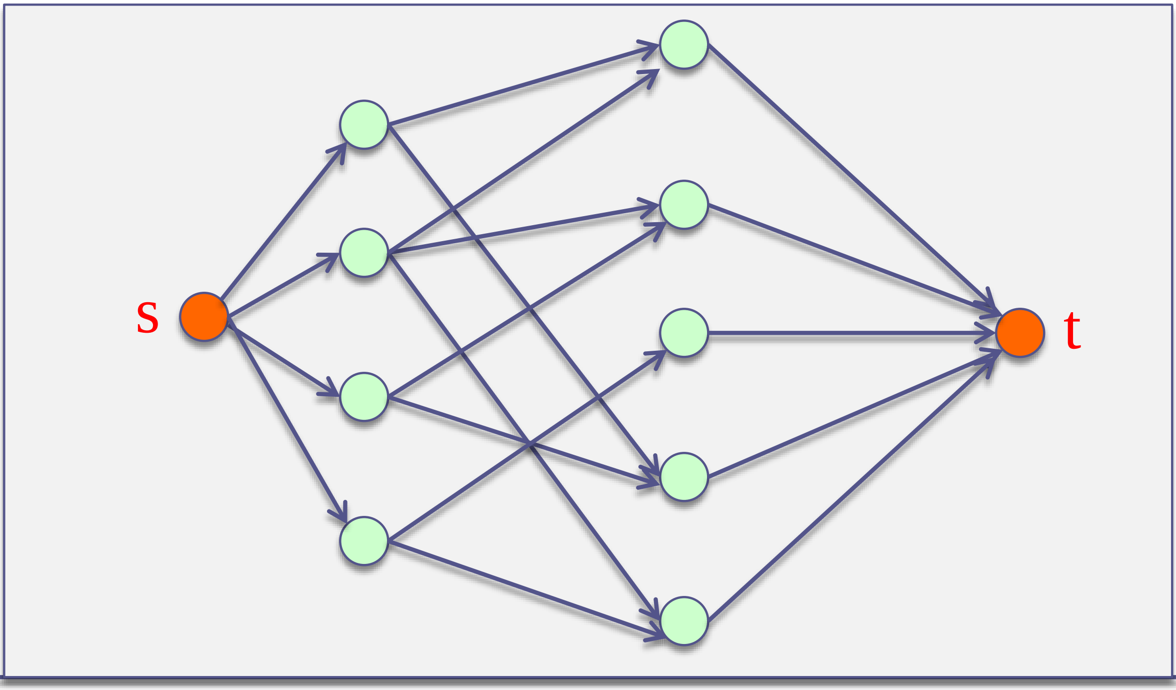
1) Set the flow value from any student node **i** to the course node **j** to be 1.
2) Set the flow value from **s** to any student node **i** to be the # of courses **i** is assigned to.
3) Set the flow value from any course node **j** to **t** to be the # of students **j** is assigned to.



Proof: (1) Assignment → Flow

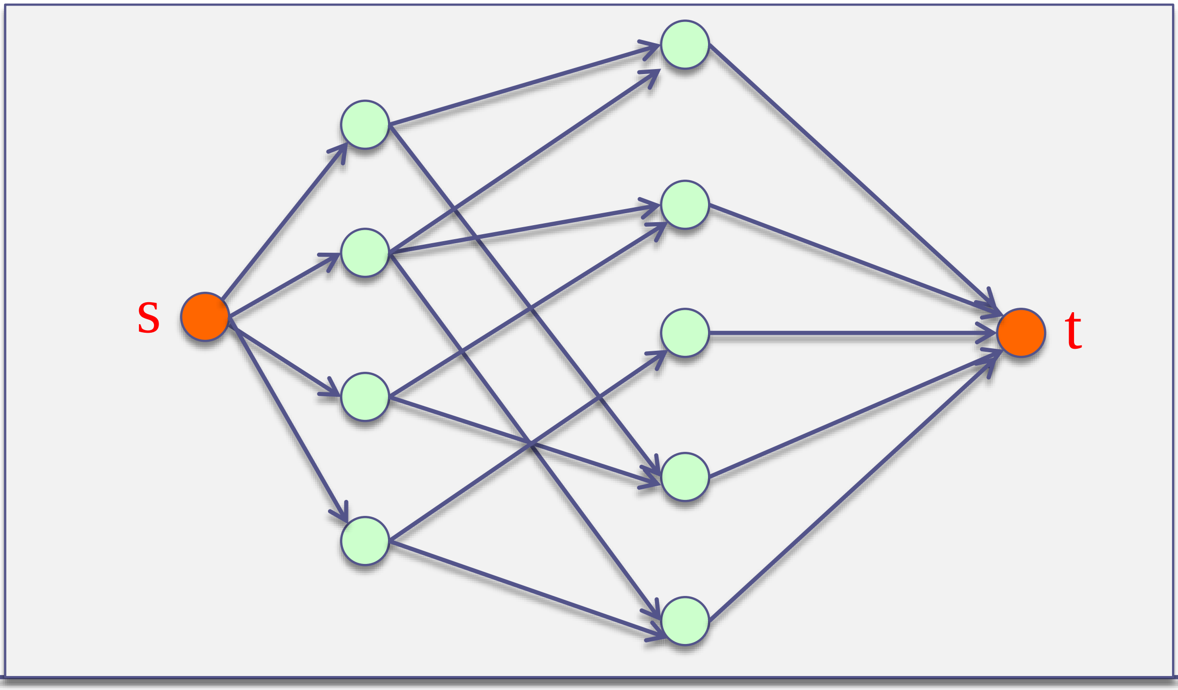
Capacity constraints:

1. Flow through and capacity of any (student → course) edge is 1
2. Student i can be assigned to at most 5 courses, and capacity(s → i) = 5
3. Course j can be assigned to at most 20 students, and capacity(j → t) = 20



Proof: (1) Assignment → Flow

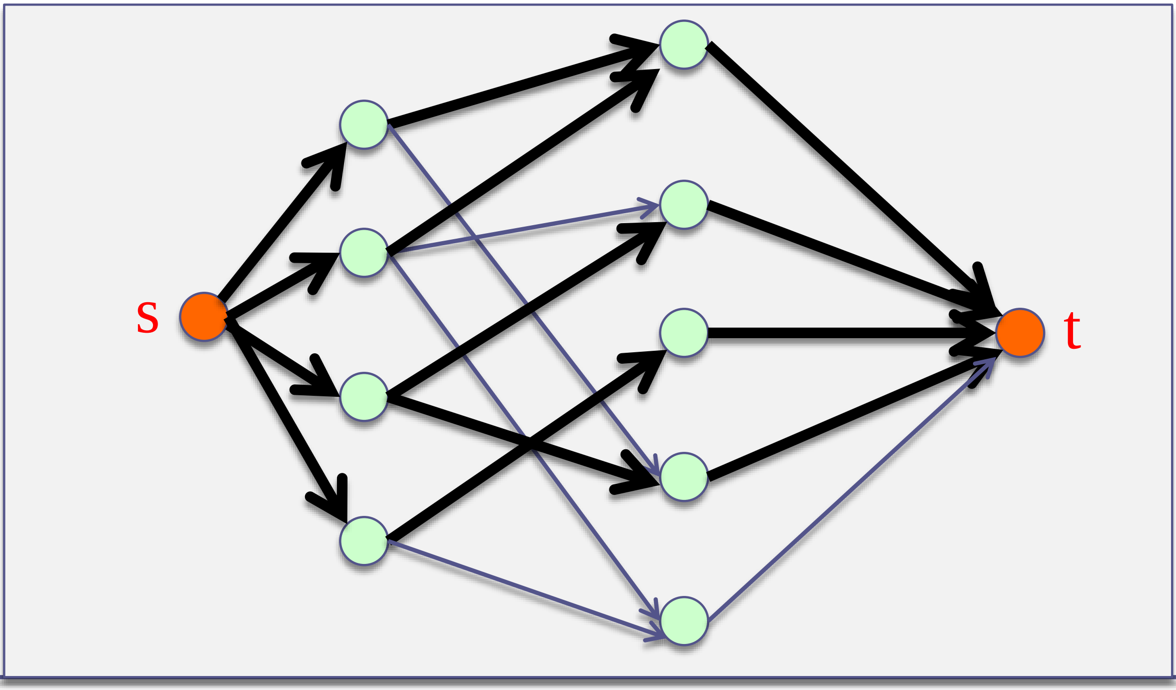
Solution size: The size of the flow going out of source s is the sum of flow values (s → i), which is equal to the # of student-to-course assignments.



Proof: (2) Flow → Assignment

(1) Show that any flow solution can be mapped to a valid assignment with the same size

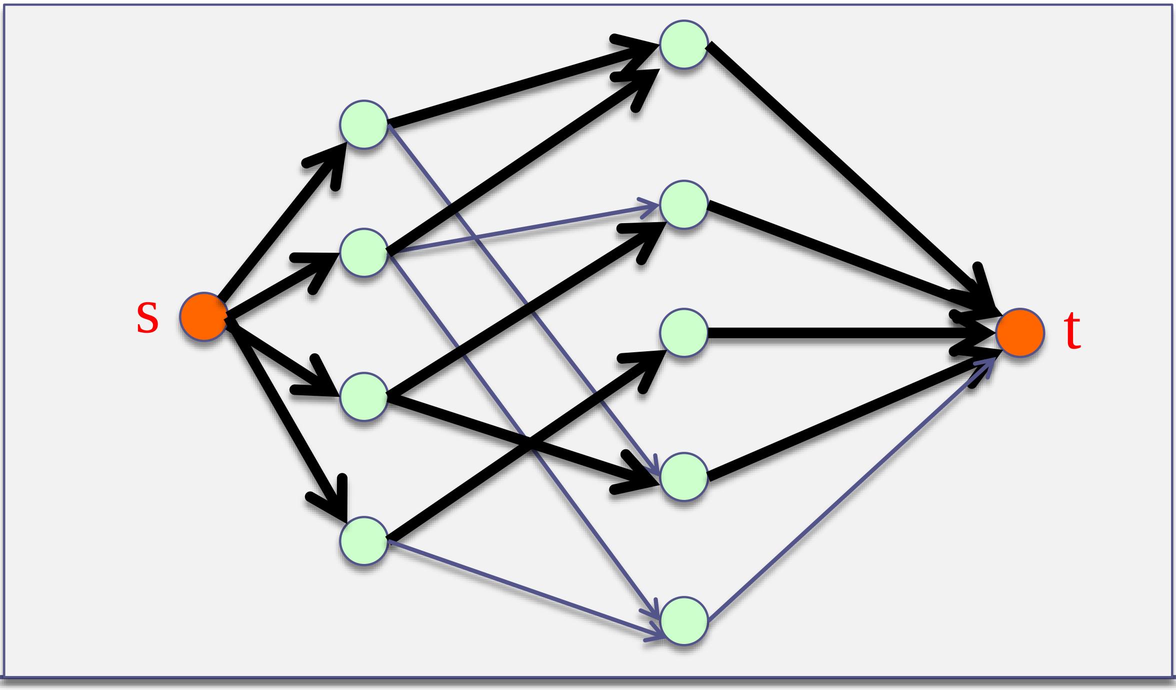
Given a flow solution of size |F|:



Proof: (2) Flow → Assignment

Create an assignment solution as described in the solution. For a solution to be valid:

1. A student is not assigned to a non-preferred course (trivial to prove)
2. A student is not assigned to more than 5 courses
3. A course is not assigned to more than 20 students



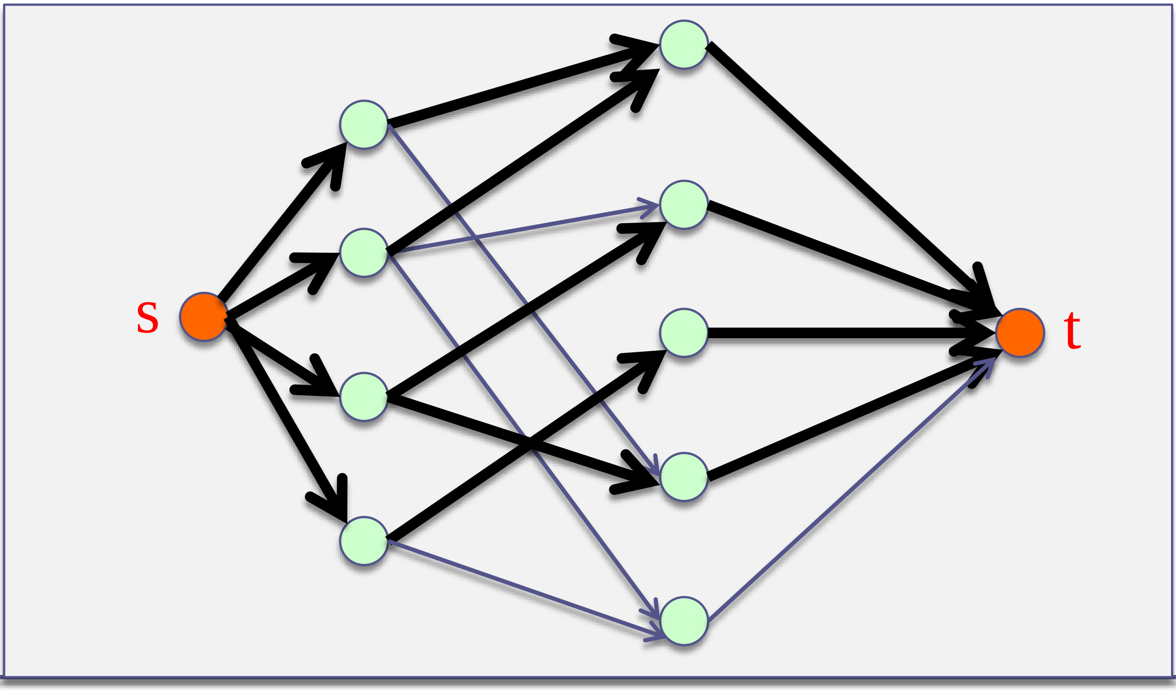
Proof: (2) Flow → Assignment

Student **i** is not assigned to more than 5 courses:

Incoming flow cannot be more than 5 (capacity constraint)

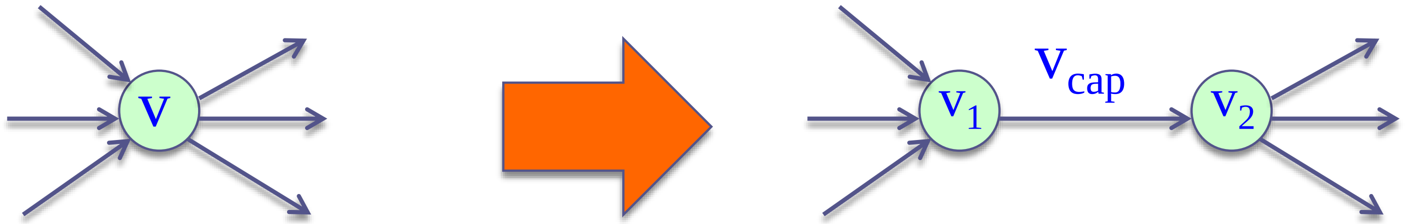
Outgoing flow cannot be more than 5 (flow conservation)

Course **j** is not assigned to more than 20 students (similar to above)



Extension for Vertex Capacities

The original flow network model can be extended to define vertex capacities.



Original vertex

Replace v with v_1 and v_2

Vertex capacity v_{cap} for vertex v can be handled by splitting v into v_1 and v_2, and setting the edge capacity between them as v_{cap}.

Min-Cost Max-Flow

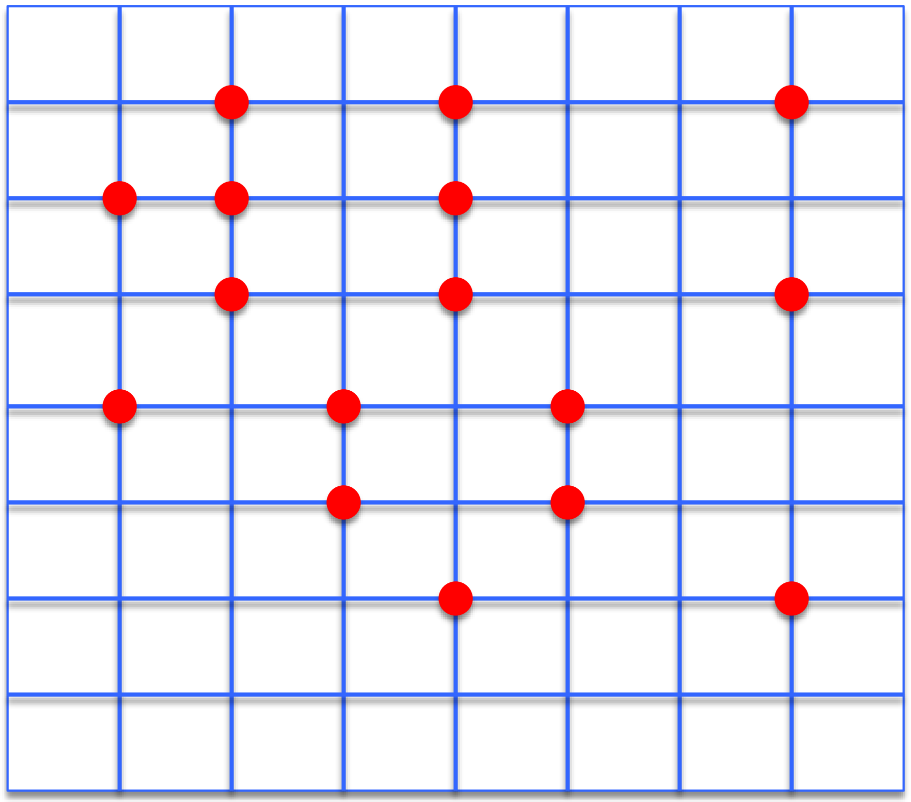
- Define a real-valued weight w for each edge e in the flow network.

- *Objective*: Compute the maximum flow in the network that minimizes the total weight W, where:

$$W = \sum_{e \in E} f(e) \cdot w(e)$$

Most max-flow algorithms can be extended easily to handle the weight minimization objective.

Exercise: Escape Routing Problem



n pins inside a chip.

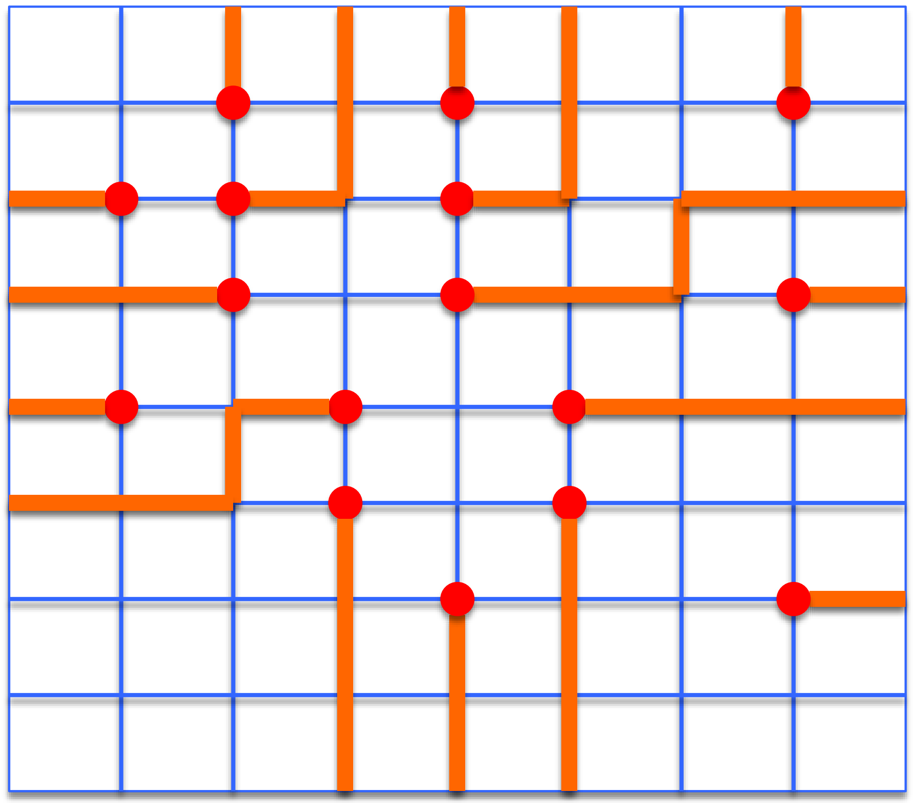
Each pin needs to be routed to a boundary point.

A routing grid is defined.

Each grid edge has a predefined routing cost.

Pins need to be routed to the boundary of the chip.

Exercise: Escape Routing Problem

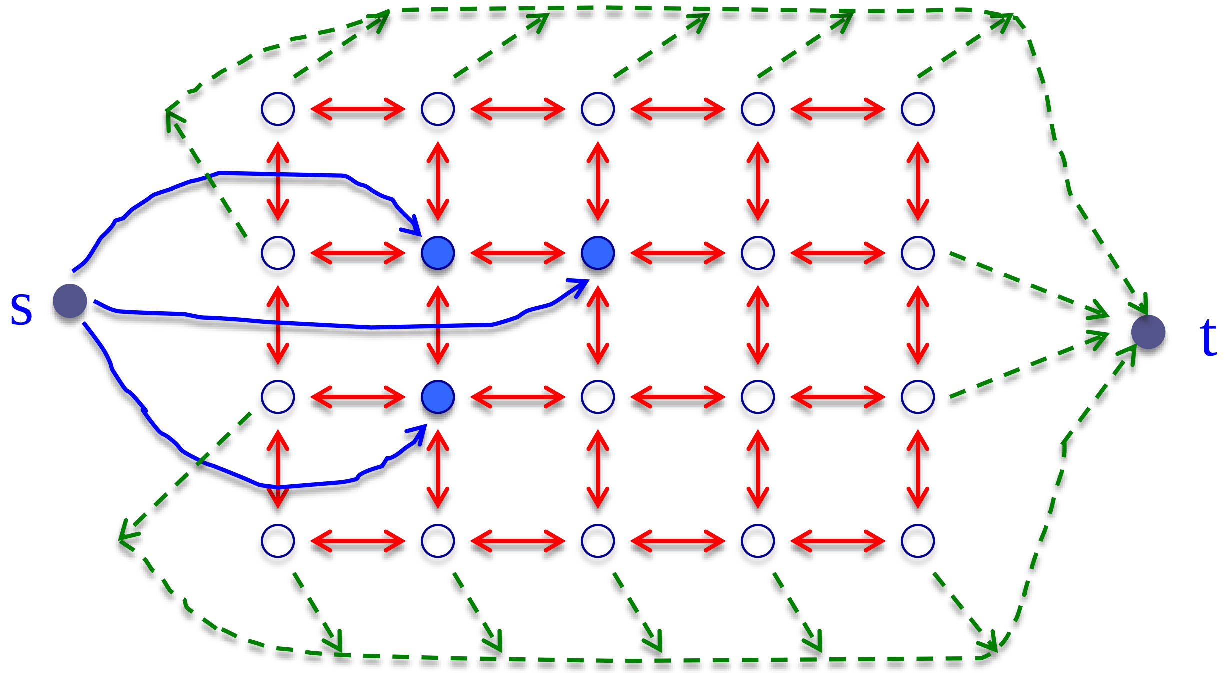


Primary objective: Route as many pins as possible to the boundary.

Secondary objective: Minimize the total edge cost.

Describe a network flow model to solve this problem.

Solution: Network Flow Model for Escape Routing



Create a vertex for each grid point with capacity 1.

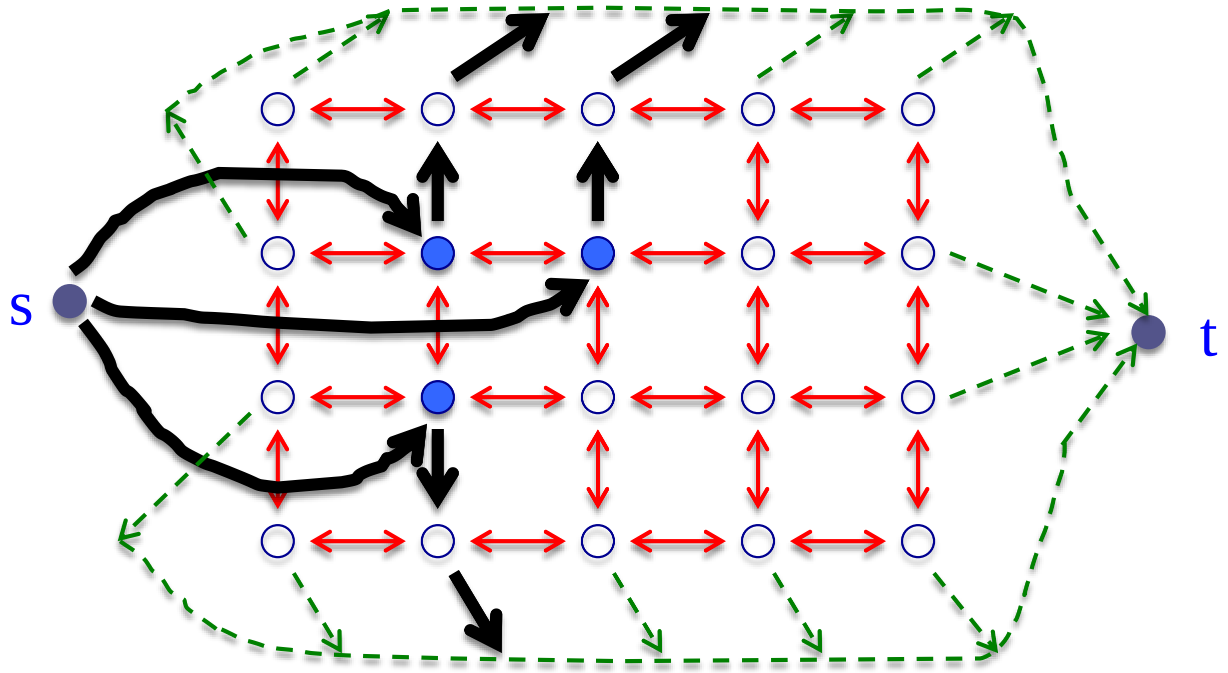
Create an edge between each neighbor grid point with the corresponding cost.

Create an edge from source s to each internal pin with zero cost.

Create an edge from each boundary pin to sink t with zero cost.

Solve min-cost max-flow.

Solution: Network Flow Model for Escape Routing



For any grid-edge with non-zero flow: create a routing edge.

All pins p with non-zero flow entering from s are routed to the boundary.

Proof: (1) Routing → Flow

(1) Show that any routing solution can be mapped to a valid flow with the same size and same cost

- Given a routing solution, create a flow network as described in the solution.
- Set the flow values as follows:
  - Set the flow of each routed grid edge to be 1
  - Set the flow of each (s, p) edge to 1 if a route starts from pin p
  - Set the flow of each (b, t) edge to 1 if a route ends at boundary point b
- Need to show that:
  - Capacity constraints are satisfied
  - Flow conservation constraints are satisfied
  - Size of flow is equal to the number of routed nets
  - Total cost of flow is equal to the total routing cost

Proof: (1) Routing → Flow

- Capacity constraints are satisfied:
  - No edge has more than 1 unit flow by definition
  - All edges have capacity of 1
- Flow conservation constraints are satisfied:
  - For any vertex corresponding to a routed pin p:
    - One unit flow enters p from s and exits through a routing edge
  - For any vertex corresponding to a routed boundary point b:
    - One unit flow enters b through a routing edge and exits to t
  - For any other vertex v that has a route passing through it:
    - One unit flow enters v through a routing edge and exits it through another routing edge.

Proof: (1) Routing → Flow

- Size of flow = # of routed nets:
  - By definition, an edge (s, p) has unit flow iff a route begins at p
  - Hence the flow size is equal to # of routed nets

- Total flow cost= total routing cost:
  - The cost of any edge (s, p) or (b, t) is zero
  - Total flow cost is the sum of edge-weight * flow
  - By definition, a unit flow passes through a grid edge iff there is a route passing through that edge.
  - Hence, total flow cost = total routing cost

Proof: (2) Flow → Routing

(2) Show that any flow solution can be mapped to a valid routing solution with the same size and the same cost

- Given a flow solution, construct routes as described in the solution.
- Need to show that:
  - A route must start at a pin p and end at a boundary point b without splitting or merging
  - No two routes can share a vertex or edge
  - The flow size is equal to the # of routed nets
  - The total routing cost is equal to the total flow cost

Proof: (2) Flow → Routing

- A route must start at a pin p and end at a boundary point b without splitting
  - From capacity constraints, an edge/vtx can have at most one unit of flow
  - Since all capacities are integers, all flow values must be integer.
  - So, flow through each edge/vtx is either 0 or 1.
  - Source s is connected to vertices corresponding to pins
  - Sink t is connected to vertices corresponding to boundary points.
  - From flow conservation, if a unit flow enters a vertex, it must exit the vertex through another edge (except for s and t).
  - Hence, each route must start from p and end at t without splitting.

Proof: (2) Flow → Routing

- No two routes can share a vertex or edge
  - From capacity constraints, an edge/vtx can have at most one unit of flow
- Flow size = # of routed nets
  - Already showed that a unit flow starting at source s corresponds to routing of one net from pin p to boundary b.
  - So, flow size is equal to the number of routed nets.
- The total routing cost is equal to the total flow cost
  - The cost of any edge (s, p) or (b, t) is zero
  - Total flow cost is the sum of edge-weight * flow
  - By definition, a unit flow passes through a grid edge iff there is a route passing through that edge.
  - Hence, total flow cost = total routing cost
- *Proof is complete.*

Multi-Commodity Flow

Define different types of flows, and enforce the conservation constraints for each flow separately.

Capacity constraints: $\sum_{i=1}^{k} f_i(u,v) \leq c(u,v)$

Flow conservation: $\sum_{w \in V} f_i(u,w) = 0 \quad \text{when} \quad u \neq s_i, t_i$

$\forall v, u, \; f_i(u,v) = -f_i(v,u)$

Optional:
Demand satisfaction: $\sum_{w \in V} f_i(s_i, w) = \sum_{w \in V} f_i(w, t_i) = d_i$

Source: Wikipedia

Multi-Commodity Flow

Max multi-commodity flow problem:

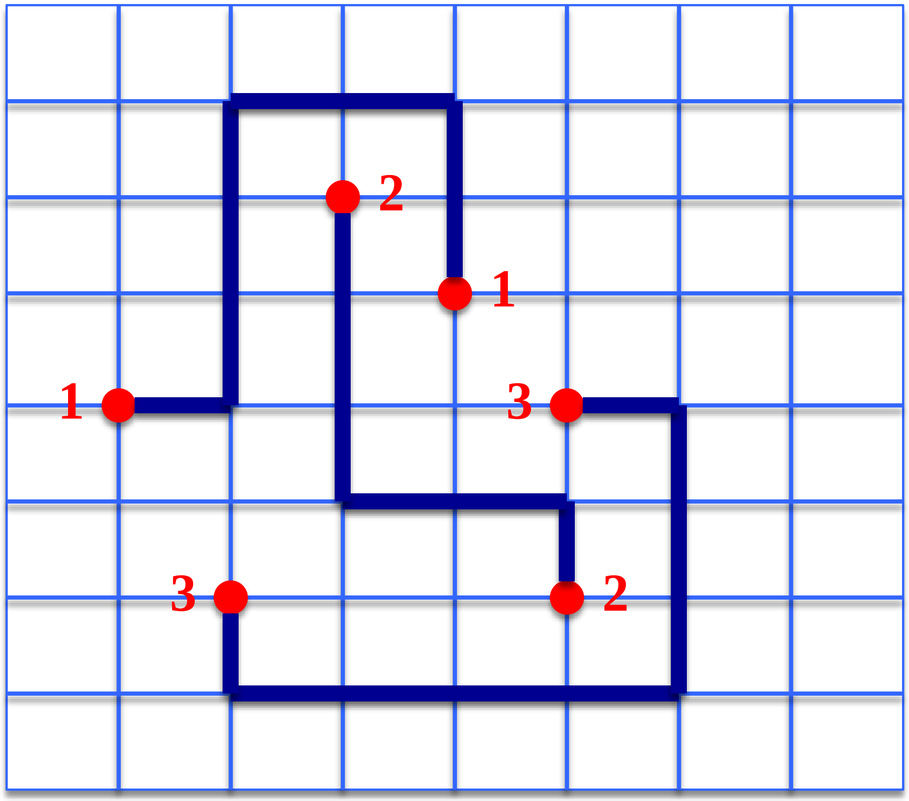
maximize $$\sum_{i=1}^{k} \sum_{w \in V} f_i(s_i, w)$$

Min-cost multi-commodity flow problem

minimize $$\sum_{(u,v)\in E} \left(a(u,v) \sum_{i=1}^{k} f_i(u,v) \right)$$

Both problems are NP-complete for integer flows.

Exercise: Single Layer Routing

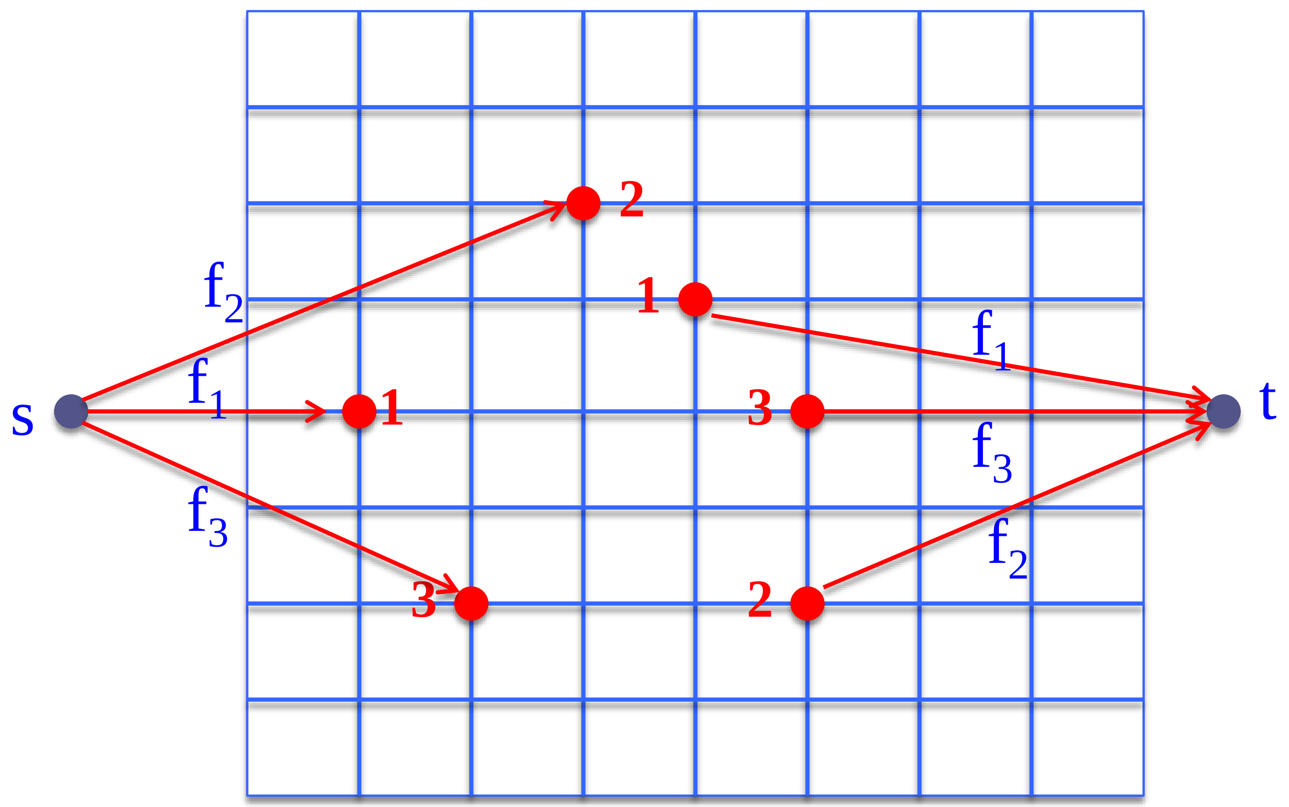


There are n 2-pin nets that need to be routed on a grid.

Objective: Route as many nets as possible and minimize the total wirelength.

Use multi-commodity network flow to model this problem.

Solution: Single Layer Routing



Grid network similar to escape routing.

Create an edge from source s to each pin i that allows only flow commodity f_i.

Create an edge from each pin i to sink t that allows only flow commodity f_i.

Solve the min-cost multi-commodity flow.

Simultaneous Pin Assignment and Routing

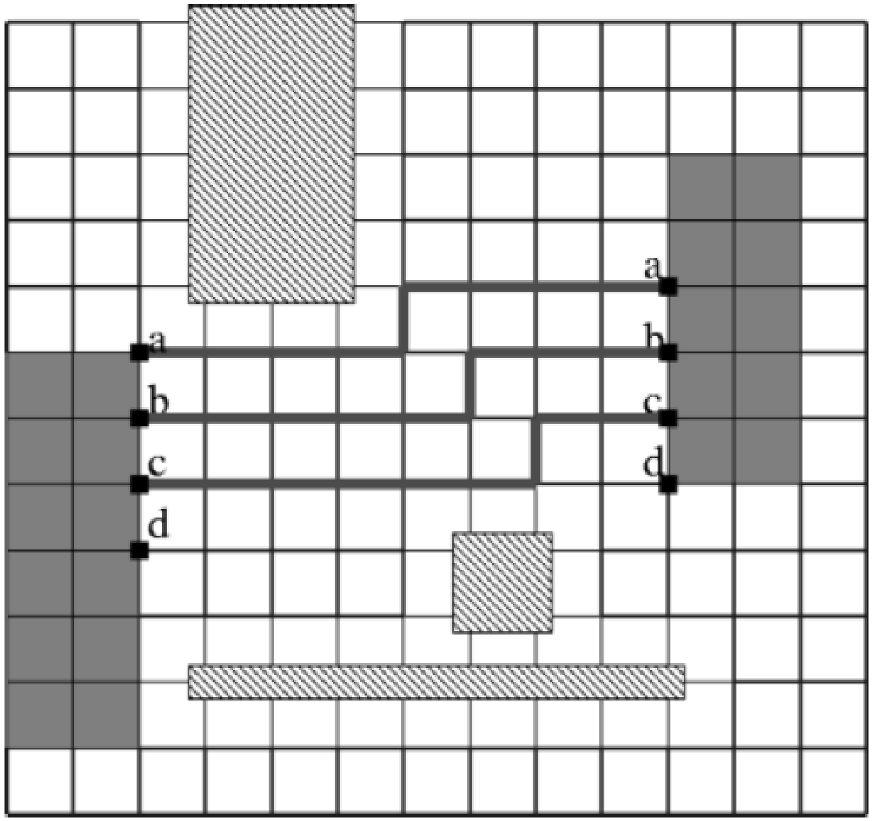
Pin assignment problem:

- Consider a macro block B with a fixed outline
- We have the netlist available for B
- We need to assign the pin locations at the block boundaries before placement

Routing between two macro blocks B_1 and B_2

- Each pin $\{a_1, b_1, c_1, \ldots\}$ of B_1 needs to be connected to the corresponding pins $\{a_2, b_2, c_2, \ldots\}$ of B_2.

Pin Assignment Followed By Routing



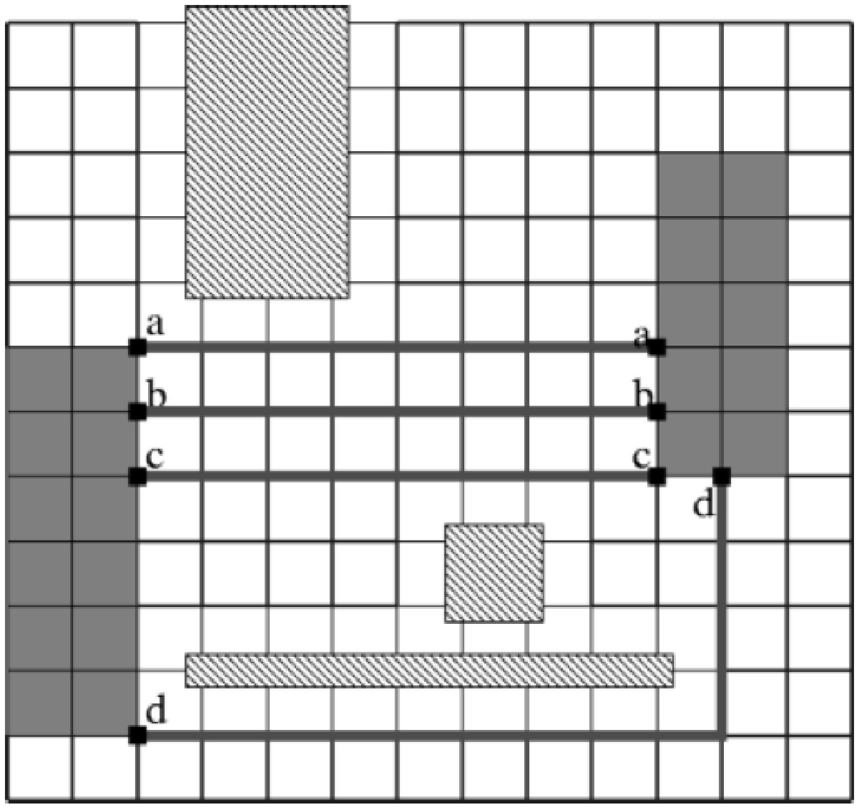
Step 1: Perform pin assignment in such a way that the pins to be connected are as close as possible.

Step 2: Perform routing between the pins.

Problem: Net d cannot be routed.

Xiang, H. et. al., "Min-Cost Flow-Based Algorithm for Simultaneous Pin Assignment and Routing", IEEE Transactions on Computer-Aided Design, Vol. 22, No. 7, July 2003

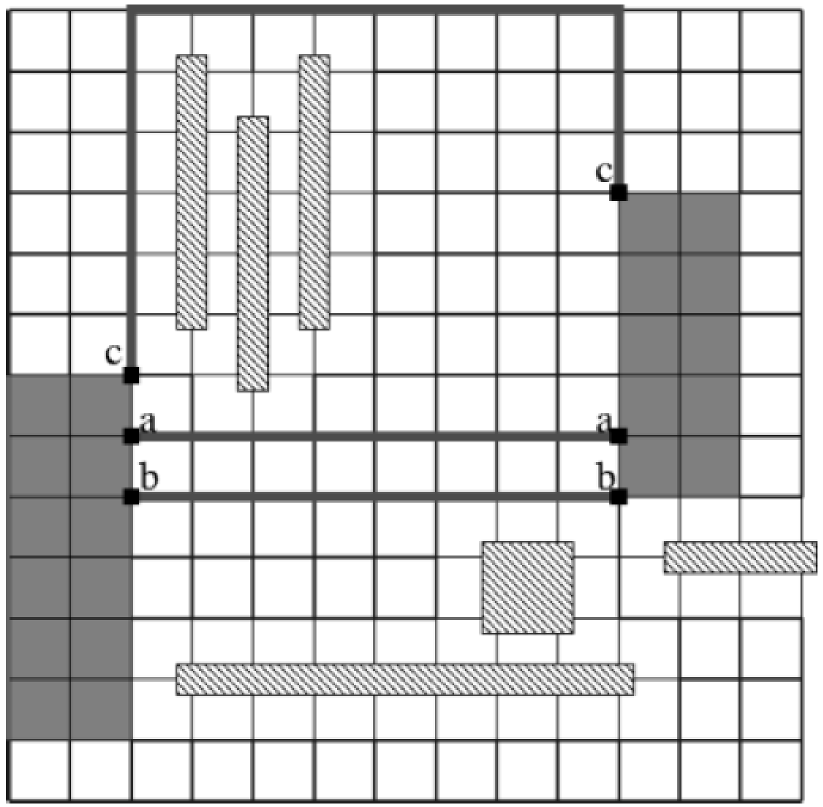
Simultaneous Pin Assignment and Routing



Simultaneous pin assignment and routing assigns all pins and routes all connections.

Xiang, H. et. al., "Min-Cost Flow-Based Algorithm for Simultaneous Pin Assignment and Routing", IEEE Transactions on Computer-Aided Design, Vol. 22, No. 7, July 2003

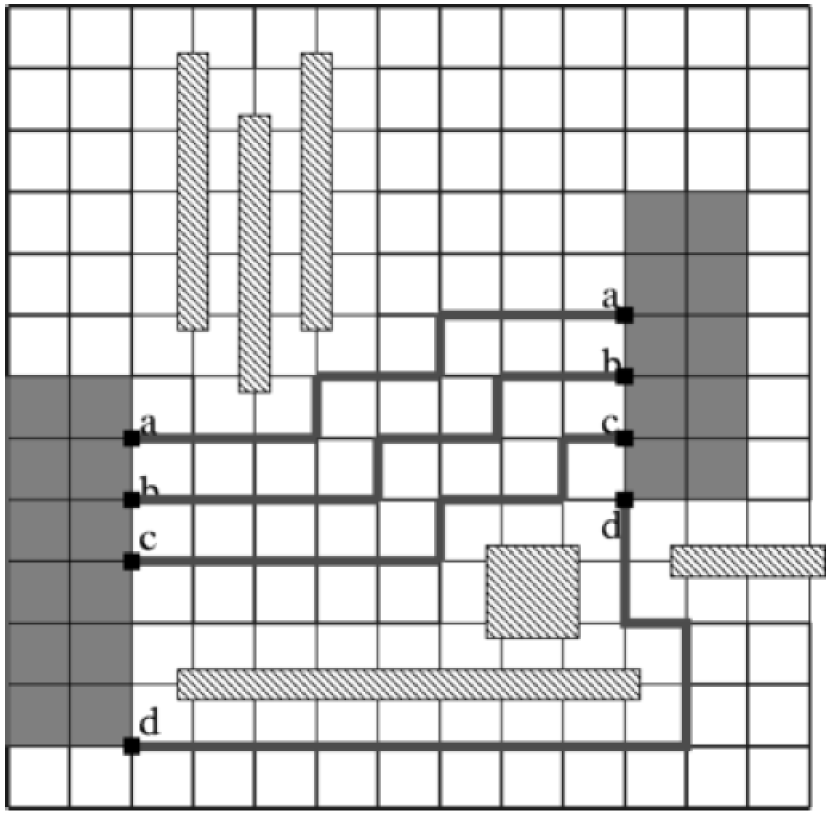
Simultaneous Pin Assignment and Routing: Greedy



Perform pin assignment and routing one net at a time in a greedy way.

Xiang, H. et. al., "Min-Cost Flow-Based Algorithm for Simultaneous Pin Assignment and Routing", IEEE Transactions on Computer-Aided Design, Vol. 22, No. 7, July 2003

Simultaneous Pin Assignment and Routing: Optimal

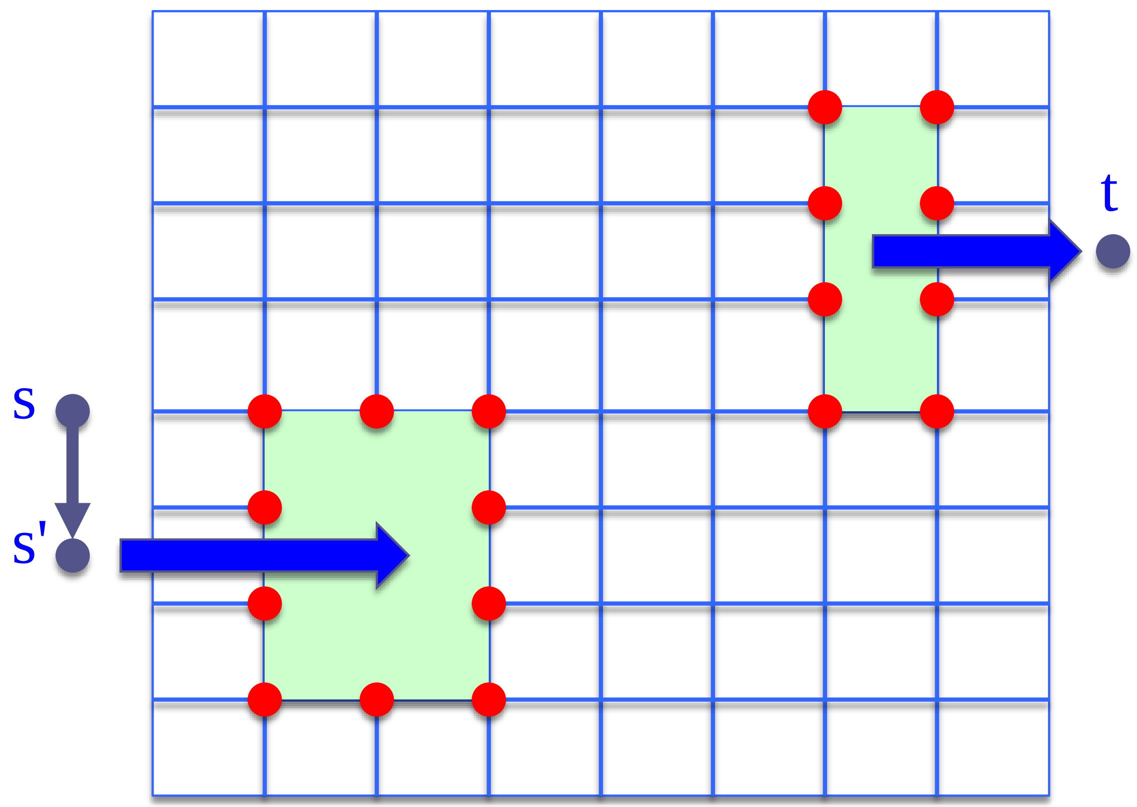


Optimal solution routes all connections with min cost.

Is there a polynomial time optimal algorithm for this problem?

Xiang, H. et. al., "Min-Cost Flow-Based Algorithm for Simultaneous Pin Assignment and Routing", IEEE Transactions on Computer-Aided Design, Vol. 22, No. 7, July 2003

Solution: Simultaneous Pin Assignment and Routing



Grid network similar to escape routing. Remove edges overlapping blockages.

Create an edge from source s to vertex s' with capacity equal to the number of pins.

Create an edge from s' to each boundary point in the first block, with capacity equal to 1.

Create an edge from each boundary point in the second block to sink t, with capacity equal to1.

Solve the min-cost max flow.