Day 5 Lecture Notes: Networks and Graph Models

Mathematical Modeling & Computational Projects Camp

Morning Structure (10:00–12:30)

- 10:00–10:30: Lecture Block 1
- 10:35–11:05: Problem Solving Session 1 (30 minutes)
- 11:10–11:40: Lecture Block 2
- 11:40–12:10: Problem Solving Session 2 (shorter) + 12:10–12:30 synthesis/review

Lecture Block 1 (10:00–10:30): Graph Models and Max Flow

We model a network as a graph $G = (V, E)$, where V is the set of vertices (nodes) and E is the set of edges (connections).

Notation and vocabulary:

- **Undirected graph**: edge $\{u, v\}$ can be used both ways.
- **Directed graph**: edge (u, v) has a direction from u to v.
- **Capacity** c_{uv}: maximum amount that can travel on edge (u, v).
- **Flow** f_{uv}: amount we choose to send on edge (u, v).

For flow problems, we use a directed graph with a **source** s (where flow starts) and a **sink** t (where flow ends).

Flow constraints:

- **Capacity constraints**: $0 \leq f_{uv} \leq c_{uv}$ for every edge.
- **Conservation constraints**: for each intermediate node $v \neq s, t$,

$$\sum_{(u,v)\in E} f_{uv} = \sum_{(v,w)\in E} f_{vw}.$$

In words: what flows into a middle node must flow out.

A **feasible** s–t **flow** is any assignment of edge flows that satisfies all capacity and conservation constraints. The **value** of a flow is total flow leaving s (equivalently entering t).

Max-flow as a linear program:

$$\max \sum_{(s,j)\in E} f_{sj}$$

subject to

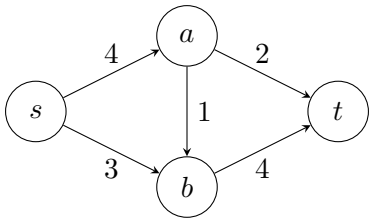
$$0 \leq f_{uv} \leq c_{uv} \quad \text{for all } (u, v) \in E,$$

$$\sum_{(u,v)\in E} f_{uv} = \sum_{(v,w)\in E} f_{vw} \quad \text{for all } v \in V \setminus \{s,t\}.$$

Worked Example 1 (small network):

- Edges and capacities: $s \to a : 4$, $s \to b : 3$, $a \to b : 1$, $a \to t : 2$, $b \to t : 4$.
- Try flow: $f_{sa} = 3$, $f_{sb} = 3$, $f_{ab} = 1$, $f_{at} = 2$, $f_{bt} = 4$.
- Check conservation: at a, inflow 3 and outflow $1 + 2 = 3$; at b, inflow $3 + 1 = 4$ and outflow 4.
- Flow value is $f_{sa} + f_{sb} = 6$.

Example flow-network sketch (capacity labels on arrows):



Since total capacity into t is $2 + 4 = 6$, this flow is optimal. So max flow $= 6$.

Worked Example 2 (high-school LP view):

- Edges: $s \to u : 5$, $u \to t : 5$, $s \to t : 2$.
- Variables: f_{su}, f_{ut}, f_{st}.
- Objective: maximize $f_{su} + f_{st}$.
- Constraints: $0 \le f_{su} \le 5$, $0 \le f_{ut} \le 5$, $0 \le f_{st} \le 2$, and conservation at u: $f_{su} = f_{ut}$.

Best choice is $f_{su} = 5$, $f_{ut} = 5$, $f_{st} = 2$, giving max flow value 7.

Problem Solving Session 1 (10:35–11:05): Practice Set A

Goal: Build and analyze flow-network models, then express max flow as an LP.

1. A school donation network has source s, sink t, and edges $s \to a : 6$, $s \to b : 4$, $a \to c : 4$, $b \to c : 2$, $a \to t : 2$, $c \to t : 6$. Draw the corresponding directed graph with capacities. Then find one feasible flow of value at least 7, or explain why that is impossible.
2. For the same network, write the max-flow LP explicitly: redraw the graph from Problem 1, then list variables, objective function, capacity constraints, and conservation constraints.
3. In a network with edges $s \to x : 5$, $s \to y : 5$, $x \to t : 3$, $y \to t : 4$, $x \to y : 2$, draw the corresponding directed graph, then compute the maximum possible flow value and justify your answer using capacities into t.
4. A student proposes flow values that violate conservation at node c (inflow 6, outflow 4). Explain in one or two sentences why this is not feasible, and give one fix.
5. Real-model question: in a package-delivery network, what might capacities represent? Give two realistic factors that could change capacities during the day.

Instructor checkpoint: verify students can separate *model structure* (who can connect to whom) from *decision variables* (how much flow to send).

Lecture Block 2 (11:10–11:40): Bipartite Graphs and Matching

A **bipartite graph** is a graph whose vertices can be split into two groups, U and V, so every edge goes between groups (never inside the same group).

Common modeling pattern:

- Left side U: people, teams, drivers, students, tasks.
- Right side V: jobs, projects, routes, mentors, time slots.
- Edge (u, v): assignment is allowed.

Matching vocabulary:

- **Matching**: set of edges with no shared endpoints.
- **Maximum matching**: matching with largest number of edges.
- **Perfect matching**: every vertex is matched (when possible).

Matching as a linear program (high-school form):

- Decision variable x_e for each edge e: $x_e = 1$ if selected, $x_e = 0$ if not.
- Objective: maximize number of selected edges,

$$\max \sum_{e \in E} x_e.$$

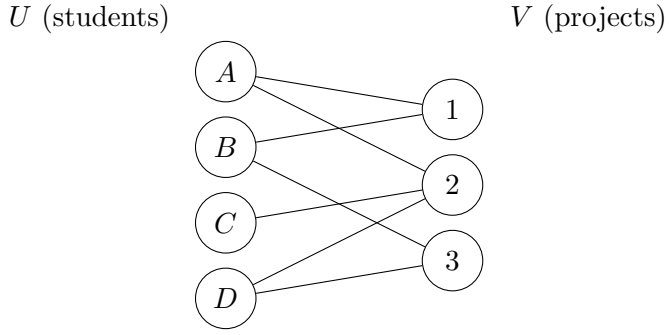
- Constraints: each vertex can be used at most once,

$$\sum_{e \ni u} x_e \leq 1 \quad (u \in U), \qquad \sum_{e \ni v} x_e \leq 1 \quad (v \in V).$$

Worked Example (student-project matching):

- $U = \{A, B, C, D\}$ (students), $V = \{1, 2, 3\}$ (projects).
- Allowed edges: A–1, A–2, B–1, B–3, C–2, D–2, D–3.
- One maximum matching is $\{A$–1, C–2, B–3$\}$ of size 3.

Example bipartite-graph sketch:



This is maximum because only three project vertices exist, so no matching can exceed size 3.

Connection to flow (high-level): we can convert a bipartite matching problem into a flow network by adding source and sink, placing capacity 1 on all edges, and computing a max flow. Each unit of flow corresponds to one matched pair.

Problem Solving Session 2 (11:40–12:10): Practice Set B

Goal: Model assignment problems with bipartite graphs and matching LPs.

1. Let $U = \{S_1, S_2, S_3, S_4\}$ and $V = \{T_1, T_2, T_3, T_4\}$. Allowed edges are S_1–T_1, S_1–T_3, S_2–T_1, S_2–T_2, S_3–T_2, S_3–T_4, S_4–T_3. Draw the corresponding bipartite graph. Then find a maximum matching and state whether a perfect matching exists.
2. Write the matching LP for Problem 1: redraw (or reuse) the graph from Problem 1, define one variable per edge, write the objective, and write all "at most one" constraints for each vertex.
3. Give a real camp example that is naturally bipartite (for example, students and project topics, or mentors and time slots). Define the two sets and what an edge means, and draw the corresponding bipartite graph.
4. Explain why the edge set $\{S_1$–$T_1,\ S_2$–$T_1\}$ is not a matching, draw the small graph containing these two edges, and modify it to become a valid matching.

Instructor move for fast finishers: ask students to create a bipartite graph with 5 vertices on each side that has maximum matching size 4, and explain why size 5 is impossible.

Synthesis and Review (12:10–12:30)

- Check one student max-flow model from Set A and one matching model from Set B.
- Revisit vocabulary: directed graph, capacity, feasible flow, bipartite graph, matching.
- Exit check: "What are the LP variables and constraints in a max-flow model?"
- Exit check 2: "What are the LP variables and constraints in a matching model?"
- Exit check 3: "How is bipartite matching connected to flow at a high level?"

Python Preview (Afternoon)

Build directed and bipartite graphs in NetworkX, compute max flow and maximum matching, and compare model outputs with hand-built LP formulations.