Graph Modeling: Network Flows to Inform Course Selection

Zakk Heile and Isaac Yang (Duke University)

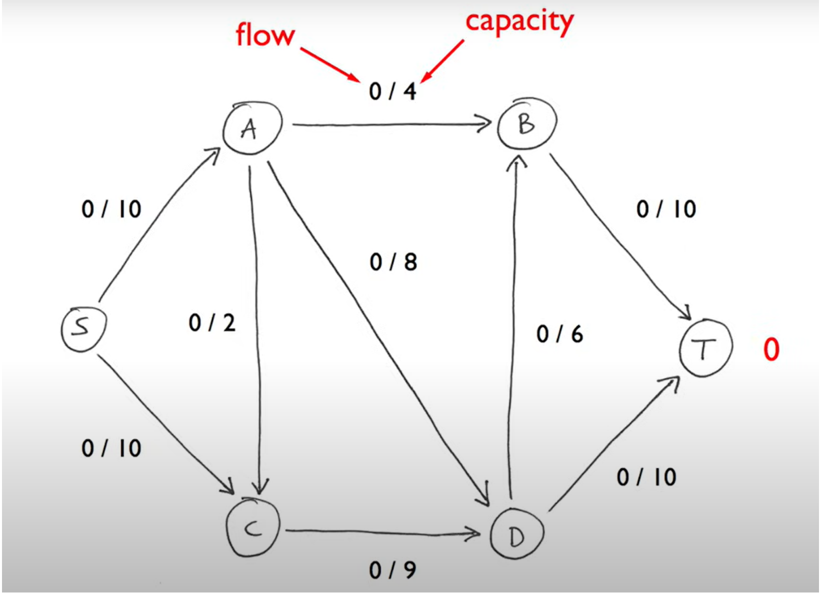
December 11, 2024

Overview of Problems

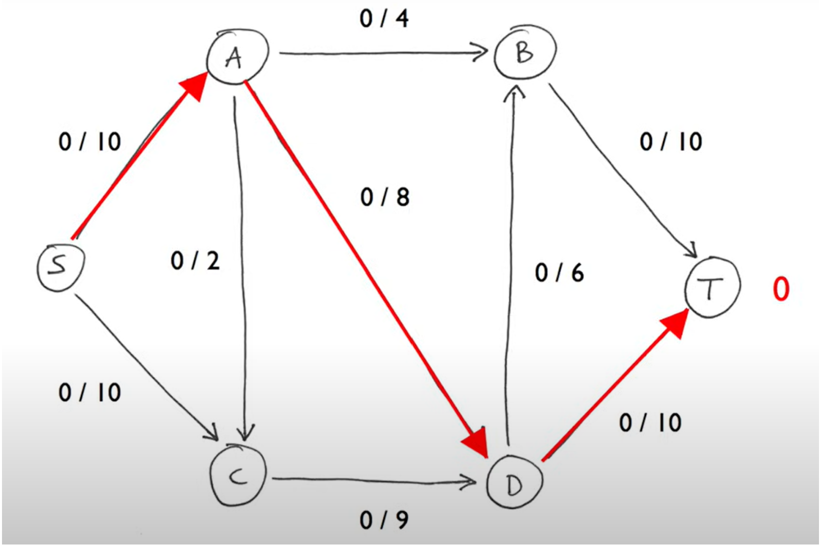
- Check if a selection of classes satisfies distribution requirements.
- Find the **minimum number** of classes to satisfy requirements.
- Optimize class selection to minimize time commitment or maximize enjoyment while satisfying requirements.
- Given an existing selection of classes, determine optimal ways to satisfy constraints.

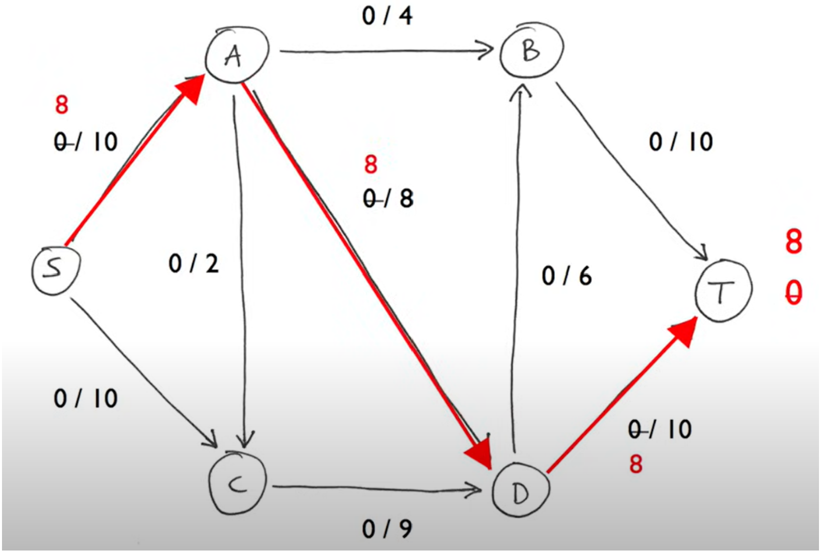
Modeling Course Selection

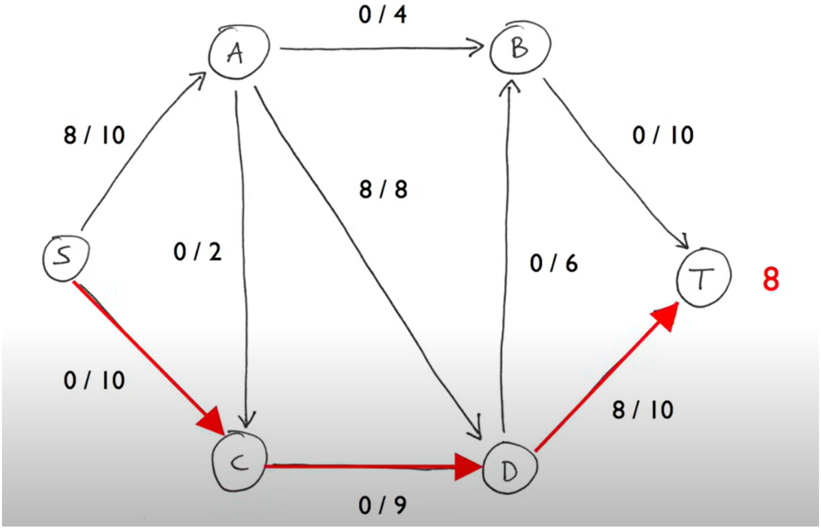
We model the Course Selection problem as a Max Flow problem.

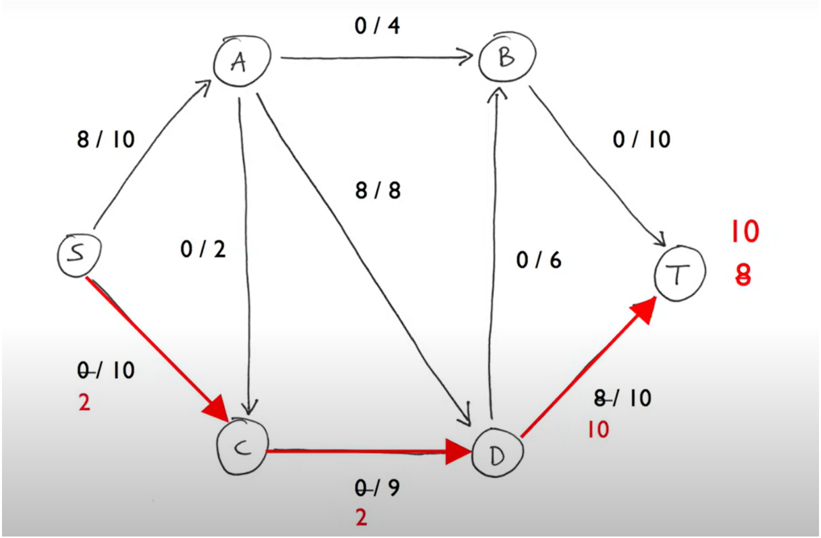


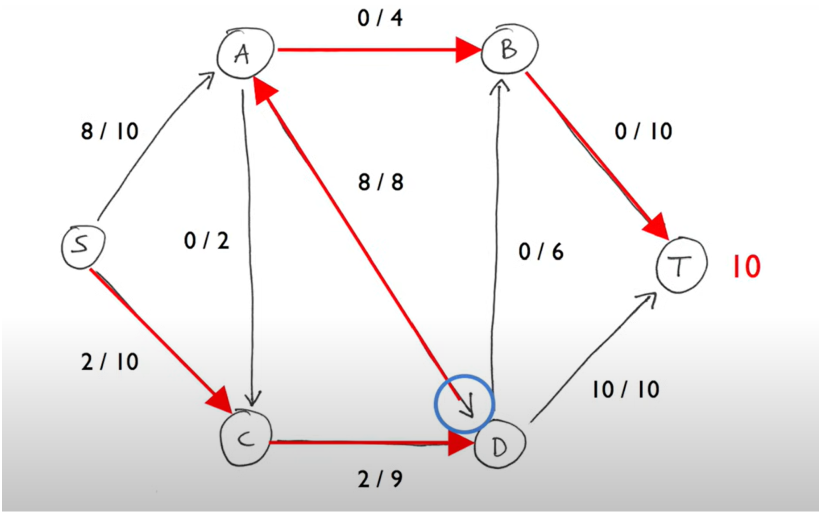
Cited from Michael Sambol

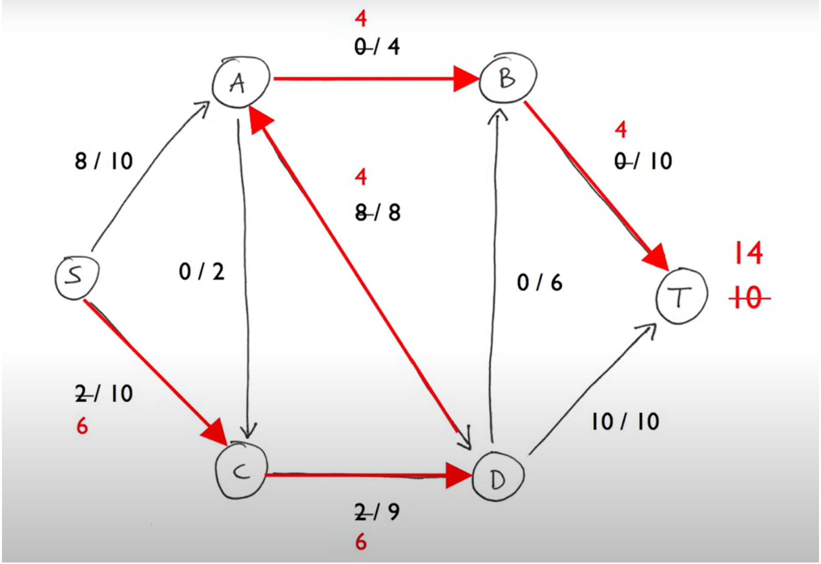

0 / 4
A
B
0 / 10
0 / 10
0 / 8
S
0 / 2
0 / 6
T
0
0 / 10
0 / 10
C
D
0 / 9


0 / 4
A
B
8
0 / 10
8
0 / 8
0 / 10
S
0 / 2
0 / 6
T
8
0
0 / 10
0 / 10
8
C
D
0 / 9


0 / 4
A
B
8 / 10
0 / 10
8 / 8
S
0 / 2
0 / 6
T
8
0 / 10
8 / 10
C
D
0 / 9


A
B
S
T
C
D
0 / 4
8 / 10
0 / 10
8 / 8
0 / 2
0 / 6
~~10~~
8
~~0~~/ 10
2
~~8~~/ 10
10
~~0~~/ 9
2

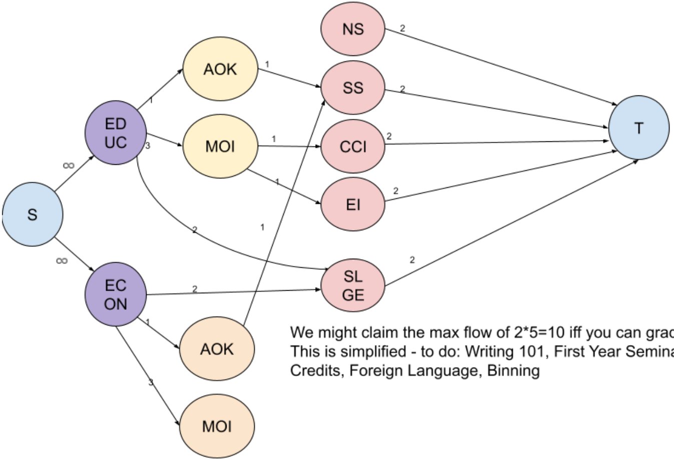

0 / 4
A
B
8 / 10
0 / 10
8 / 8
S
0 / 2
0 / 6
T
10
2 / 10
10 / 10
C
D
2 / 9



(This is not a max flow, more steps to do but we leave it here). Time complexity: $O(f \cdot E)$ or $O(V \cdot E)$

Architecture

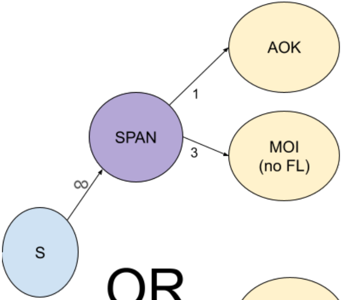
- **Source Node**: Connects to class nodes (infinite capacity).
- **Class Nodes**: Link to specific general requirement nodes (specific to the course):
  - Modes of Inquiry
  - Areas of Knowledge
  - Seminar Requirements
- **Requirement Nodes**: Flow general requirement nodes to shared nodes (e.g., SS, CCI).
- **Sink Node**: Collects all flow to confirm requirement satisfaction.



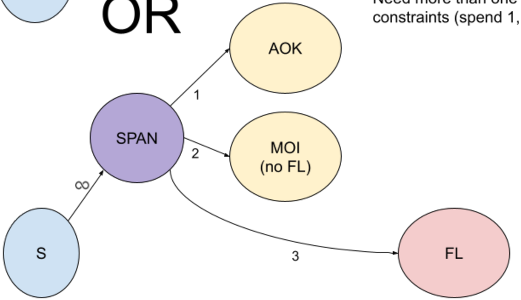
We might claim the max flow of 2*5=10 iff you can graduate.
This is simplified - to do: Writing 101, First Year Seminar, ½ Credits, Foreign Language, Binning

Constraints

- Classes may fulfill multiple Modes of Inquiry (up to 3) or Areas of Knowledge (1).
- Given that courses may be coded with more than that amount, many allocations are possible for given classes, complicating flow conservation.
- **Foreign Languages**: Special rules:
  - 3 courses at 100/200 level, or
  - 1 course at 300+ level, which satisfies all 3 requirements.
- Complex conservation rules:
  - A 300+ level foreign language course fulfills the Foreign Language requirement with weight 3 without penalizing other Modes of Inquiry or Areas of Knowledge.



OR



Problem:

Can't have both gadgets in the same graph because a max flow algorithm could run partial flows to each gadget

Need more than one gadget because of conservation constraints (spend 1, get 3 free for 300-level FL code)

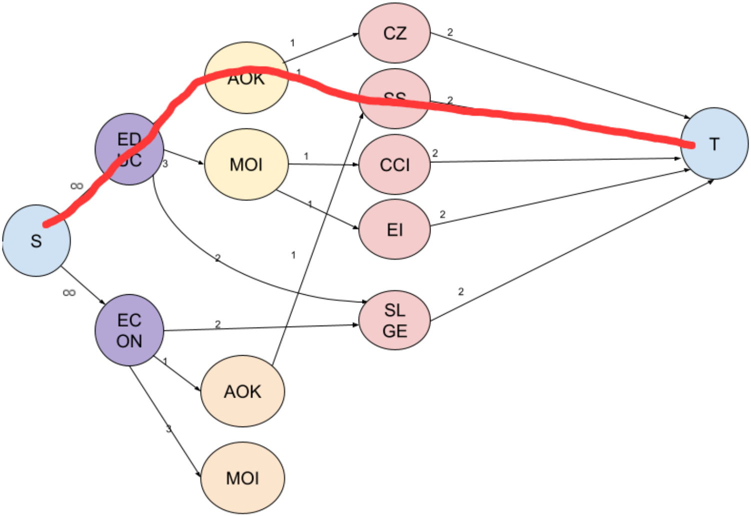
Binning

- Many classes share the same codes, increasing the complexity of graph construction and max flow computation.
- Solution: Bin classes into sets of codes, focusing on the sets needed to satisfy requirements.
- A set of codes may be used multiple times (up to the number of available classes).
- Additional upper bounds for code set usage:
  - $2 \times \text{numNonFLcodes} + 3 \times \text{ifFLCode}$
  - 12 (I can construct 12 courses to graduate)
- How many copies? Minimum of all 3 of those bounds.

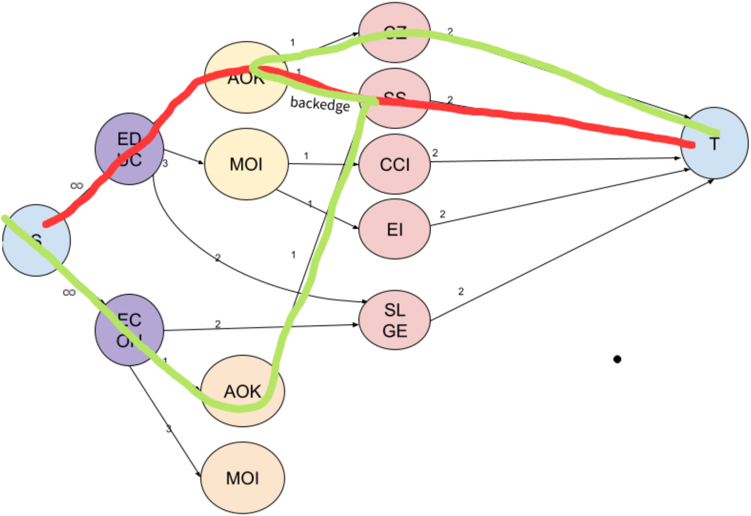
Handling Already-Taken Classes

- Codes used from already-taken classes are not fixed but can be assigned in multiple ways.
- Assign codes in all possible ways for the given classes:
  - Possibilities for a class: $\binom{4}{3}$ or $\binom{2}{1}$, or rarely their product.
- For each assignment:
  - Build a graph, push flow through the assigned codes.
  - Remove the used flow and reduce capacity accordingly.
  - Decrease the remaining max flow required for graduation.
- Evaluate all graphs and max flows and select the best.

Handling Already-Taken Classes Efficiently



Idea: restrict valid s-to-t paths.



This path is allowed, but backtracking to the course is not.

Enumerating All Max Flows

- During max flow computation, at each iteration:
  - Identify all possible s-to-t augmenting paths.
  - For each path, branch into a new state by pushing flow along that path.
- Recursively explore all branches, maintaining flow conservation and capacity constraints.
- Keep only the unique max flow distributions.
- Note: smartly choosing s-to-t paths can improve efficiency. For example, when there is still a path to get more codes on a class already locked in, don't explore new classes.

Finding All Maximum Flows Elegantly

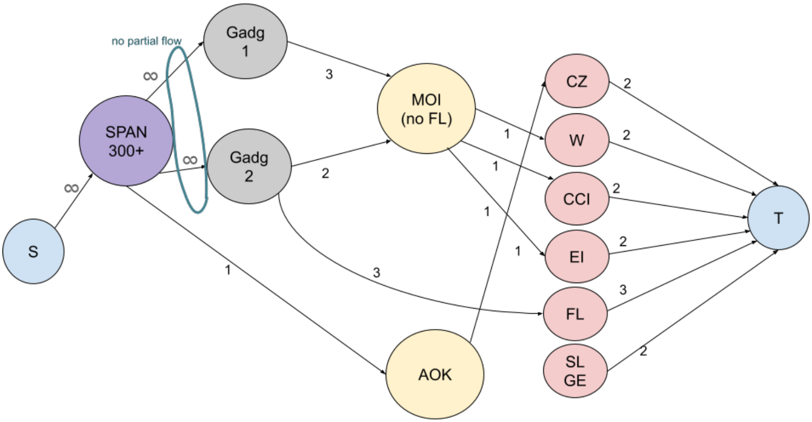
- Given two maximum flows f_1 and f_2:
  - $f' = f_2 - f_1$ is a circulation ($\sum_{u \in V} f'_{uv} - \sum_{u \in V} f'_{vu} = 0$).
- Thus, any two max flows differ by a circulation!
- Thus, if we have a max flow and can find all circulations, we can get every other max flow.
- Non-trivial circulations correspond to cycles in the residual graph:

Finding Max Flows with Minimum Classes

- Goal: Identify maximum flows that use the smallest number of classes.
- If we compute all the max flows then we just need to filter out the ones using more than the minimum number of classes.
- This approach is computational, but there is no easy way to find all max flows subject to a constraint without it (that we can come up with).
- We want all max flows (specifically with bins) because there are not that many (if our upper bound on bin copies is tight) and the user may have a preference (I would!).

Revisiting Foreign Language Gadgets

- **Key Insight:** If we compute all max flows, we can handle the Foreign Language (FL) constraints flexibly.
- **Strategy for 300 Level $\leq$ 2nd class:**
  - Include both FL gadgets:
    - Capacity 3 with no FL option.
    - Capacity 2 MOI no FL with Capacity 3 FL.
- **Process:**
  1. Compute all possible max flows.
  2. Filter out flows that:
     - Send partial flow to both gadgets.
  3. Minimize objectives or return among the valid flows.



We want to allow backtracking and switching between gadgets, unlike taken courses.

Conclusion

- Network flow models provide a powerful framework to optimize course selection even with obtuse constraints.
- Future work will focus on refining optimization techniques and algorithm efficiency.

Contact Information:

- Zakk Heile, Duke '27: zakk.heile@duke.edu
- Isaac Yang, Duke '26: isaac.y@dukekunshan.edu.cn