COMPSCI 311: Introduction to Algorithms

Lecture 19: Network Flow Applications

Marius Minea

University of Massachusetts Amherst

slides credit: Dan Sheldon

3 April 2019

Review: Network Flow Properties

- $v(f) \leq c(A, B)$ for any flow f and any s-t cut $c(A, B)$
  (flow value lemma)
- On termination, Ford-Fulkerson defines an $s-t$ *cut* (A, B)
  in the residual graph ($A =$ all nodes reachable from s)
  all edges *out of* A across the cut are saturated
  all edges *into* A across the cut carry no flow
- Max flow equals min cut
  Ford-Fulkerson gives a max flow, and (A, B) is a min-cut.

Complexity

- Basic F-F: $O(mnC_{\max})$ **pseudo-polynomial**
- Capacity-scaling: $O(m^2 \log C_{\max})$ **polynomial**
- Edmonds-Karp $O(m^2 n)$, Dinitz $O(mn^2)$ **strongly-polynomial**

Network Flow Application: Matching in Graphs

- In an undirected graph $G = (V, e)$, a set of edges $M \subseteq E$ is a matching if no two edges in M have a common node.
- The maximum matching problem is to find the matching with the most edges.
- How to find *some* matching?
  - choose an edge
  - delete all edges with common endpoints
  - repeat

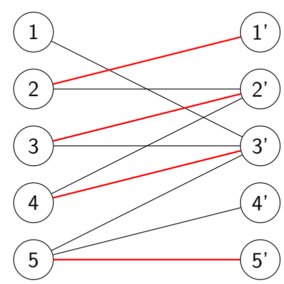
When done, matching cannot be extended, i.e., *maximal*
(different from *maximum*: highest number).

- We'll design an efficient algorithm for maximum matching in a **bipartite** graph.

Bipartite Matching

Recall: A graph is **bipartite** if we can partition nodes in two sets L and R, so each edge connects a node in L with a node in R.

Problem: Given bipartite graph $G = (L \cup R, E)$, find a maximum cardinality matching.

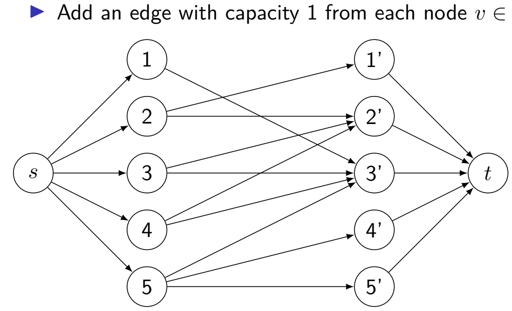


Formulating Matching as Network Flow problem

- **Goal**: given matching instance $G = (L \cup R, E)$:
  - create a flow network G',
  - find a maximum flow f in G'
  - use f to construct a maximum matching M in G.
- Exercise
- Convert undirected bipartite graph G to flow network G'
  - Direction of edges?
  - Capacities?
  - Source and sink?

Maximal Matching as Network Flow

- Add a source s and sink t
- For each edge $(u, v) \in E$, add $u \to v$ (directed), capacity 1
- Add an edge with capacity 1 from s to each node $u \in L$
- Add an edge with capacity 1 from each node $v \in R$ to t.



Clicker Question

Let G' be the flow network as constructed above and let e be an edge from L to R.

A. For every flow f, either $f(e) = 0$ or $f(e) = 1$.

B. For every maximum flow f, either $f(e) = 0$ or $f(e) = 1$.

C. There is some maximum flow f such that either $f(e) = 0$ or $f(e) = 1$.

D. B and C

E. A, B, and C

Maximum Matching: Analysis

- Run F-F to get an integral max-flow f
- Set M to the set of edges from L to R with flow $f(e) = 1$
- Claim: The set M is a maximum matching.

Correctness: We will show that:

- for every integer flow of value k there is a matching M of size k
- and vice versa.

Therefore, a maximum integer-valued flow yields a maximum matching.

Correctness 1

1. Integral flow f of value $k \Rightarrow$ matching M of size k

- Suppose f is a flow of value k
- Let $M =$ edges from L to R carrying one unit of flow
- There are k such edges, because the net flow across cut between L and R is k, and there are no edges from R to L
- There is at most 1 unit of flow entering $u \in L$, and therefore at most 1 unit of flow leaving u
- Since all flow values are 0 or 1, this means M has at most one edge incident to u.
- A similar argument for $v \in L$ means that M has at most one edge incident to v
- Therefore, M is a matching with size k

Correctness 2

2. Matching M of size $k \Rightarrow$ integral flow f of value k

- Suppose M is a matching of size k
- Send one unit of flow from s to $u \in L$ if u is matched
- Send one unit of flow from $v \in R$ to t if t is matched
- Sent one unit of flow on e if e is in M
- All other edge flow values are zero
- Verify that capacity and flow conservation constraints are satisfied, and that $v(f) = k$.

Clicker Question

What is the running time of the Ford-Fulkerson algorithm to find a maximum matching in a bipartite graph with $|L| = |R| = n$? (Assume each node has at least one incident edge)

A. $O(m + n)$

B. $O(mn)$

C. $O(mn^2)$

D. $O(m^2n)$

Perfect Matchings in Bipartite Graphs

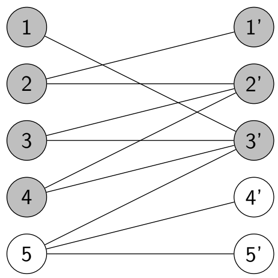
Recall: A matching M is **perfect** if every node appears in (exactly) one edge in M.

Question: When does a bipartite graph have a perfect matching?

- Clearly, we must have $|L| = |R|$
- Clearly, every node must have at least one edge
- What other conditions are necessary? Sufficient?

Perfect Matchings in Bipartite Graphs

For $S \subseteq L$, let $N(S) \subseteq R$ be the set of all neighbors of nodes in S



Observation: For a perfect matching we need

$$\forall S \subseteq L, \quad |N(S)| \geq |S| \qquad (*)$$

Otherwise we can't match all nodes in S

Hall's Marriage Theorem

Assume G is bipartite with $|L| = |R| = n$.

Simple Observation: If G has a perfect matching then:

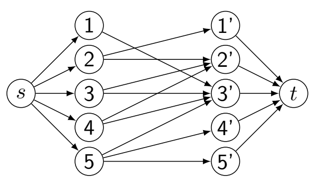
$$\forall S \subseteq L, \quad |N(S)| \geq |S| \qquad (*)$$

Theorem (Hall 1935, earlier by Frobenius, Kőnig):
G has a perfect matching **if and only if** (*)

We will prove:
if G does not have a perfect matching then (*) does not hold
$\implies$ there is some $S \subseteq L$ with $|N(S)| < |S|$.

Use max-flow/min-cut theorem on bipartite-matching flow network.

Clicker Question



Consider the flow network construction for bipartite matching.
Which of the following is true?

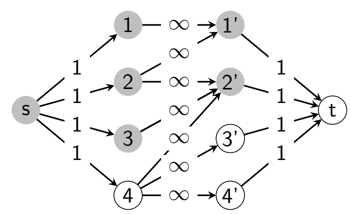
A. The construction still works if edges from s to L have infinite capacity.

B. The construction still works if edges from L to R have infinite capacity.

C. The construction still works if edges from R to t have infinite capacity.

Hall's Marriage Theorem

- Suppose G does not have a perfect matching
- Let (A, B) be the minimum-cut in G' $\implies c(A, B) < n$
- Let $S = A \cap L$.
- Then $N(S) \subseteq A$, else we cut an edge with $c(e) = \infty$
  $\implies N(S) \subseteq A \cap R$



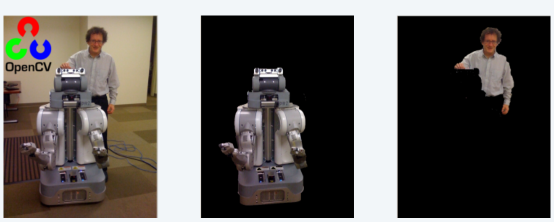
$$\begin{aligned} n > c(A, B) &= |B \cap L| + |A \cap R| \\ &= n - |S| + |A \cap R| \\ &\geq n - |S| + |N(S)| \end{aligned}$$

- $\implies |S| > |N(S)|$

Image segmentation

Image segmentation.
- Divide image into coherent regions.
- Central problem in image processing.

Ex. Separate human and robot from background scene.



Grabcut image segmentation

Grabcut. [Rother–Kolmogorov–Blake 2004]


"GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts

Carsten Rother* Vladimir Kolmogorov† Andrew Blake‡
Microsoft Research Cambridge, UK

Figure 1: **Three examples of GrabCut .** The user drags a rectangle loosely around an object. The object is then extracted automatically.

Bokeh Effect: Blurring Background

- Using an expensive camera and appropriate lenses, you can get a “bokeh" effect on portrait photos:
  the background is blurred and the foreground is in focus.

- Can fake effect using cheap phone cameras and appropriate software

Formulating the Problem

Given set V of pixels, classify each as foreground or background. Assume you have:

- Likelihood that a pixel is in foreground (a_i) / background (b_i)
- Numeric penalty p_{ij} for assigning neighboring pixels i and j to different classes

Graph edges E: for each pixel, edge to neighbors (4? 8? other?)

Criteria:

- Accuracy if $a_i > b_i$, would prefer to label pixel i as foreground
- Smoothness: if many neighbors are labeled the same (foreground), would like to label pixel i as foreground (minimize penalties)

Image Segmentation as Network Flow

Maximize correct labeling scores, minimize penalties

Let A: set of pixels labeled foreground, B: pixels in background

Maximize: $\sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{(i,j) \in E, i \in A, j \in B} p_{ij}$

Insight: (A, B) is a partition $\Rightarrow$ forms a **cut**

First sum is $\sum_{i \in V}(a_i + b_i) - \sum_{i \in A} b_i - \sum_{j \in B} a_j$
(constant minus “penalties” for mislabeling)

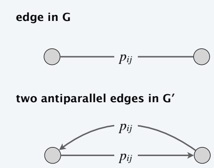
Must **minimize** $\sum_{i \in A} b_i + \sum_{j \in B} a_j + \sum_{(i,j) \in E, i \in A, j \in B} p_{ij}$

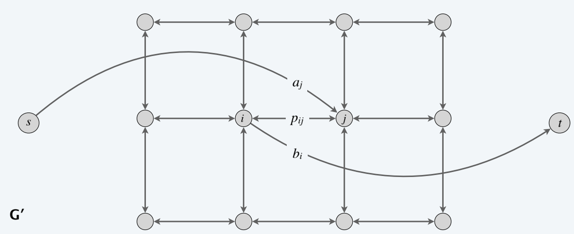
$\Rightarrow$ find **minimum cut**

Image segmentation

Formulate as min-cut problem $G' = (V', E')$.

- Include node for each pixel.
- Use two antiparallel edges instead of undirected edge.
- Add source s to correspond to foreground.
- Add sink t to correspond to background.





slide credit: Kevin Wayne / Pearson

Image segmentation

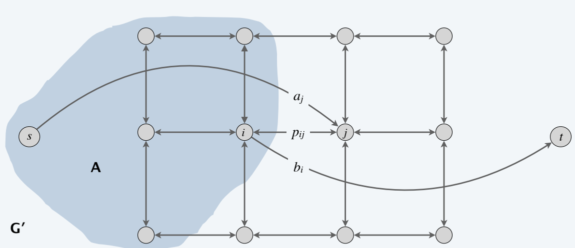
Consider min cut (A, B) in G'.

- A = foreground.

$$cap(A,B) = \sum_{j \in B} a_j + \sum_{i \in A} b_i + \sum_{\substack{(i,j) \in E \\ i \in A,\ j \in B}} p_{ij}$$

if i and j on different sides, p_{ij} counted exactly once

- Precisely the quantity we want to minimize.



slide credit: Kevin Wayne / Pearson

More Network Flows

- Extensions
  - Multiple sources and sinks
  - Circulations with supplies and demands
  - Flows with lower bounds
- Improved Algorithms: Preflow-push $O(n^3)$
- Applications
  - Network connectivity
  - Data mining: survey design
  - Airline scheduling
  - Baseball elimination
  - Multi-camera placement / scene reconstruction