CS 420: Advanced Algorithm Design and Analysis
Spring 2015 – Lecture 16

Department of Computer Science
University of British Columbia

March 05, 2015

Announcements

Assignments...

- Asst6/7...out today (due March 19)

Midterm III...

- Q/A session...March 24; 5:30-7:00; DMPT 110
- Exam...March 25; 5:30-7:00; DMPT 110
- ...on *all* course material up to and including March 19 lecture

Announcements (cont.)

Readings...

- matchings and network flows [Kleinberg&Tardos, Chapt. 7], [Cormen et al., Chapt. 26], [Dasgupta et al., Chapter 7]
- reductions and NP-hardness [Kleinberg&Tardos, Chapt. 8, 11], [Cormen et al., Chapt. 34,35]

Last class...

Matchings and Network Flows

- matchings
  - definitions
  - bounds for bipartite matchings
  - duality

Matchings in Bipartite Graphs

Since $|M| \leq |V_L \backslash S| + |N(S)|$ holds for all matchings M and all sets $S \subseteq V_L$, it follows that:

Claim:

$$\max_{\text{matchings } M} |M| \leq \min_{S \subseteq V_L} \{|V_L \backslash S| + |N(S)|\}$$

Note: this holds even if edges can be chosen *fractionally*

Matchings in Bipartite Graphs – Berge's Theorem

Suppose that M does not admit an augmenting path.

Let S_M denote the set of vertices in $v \in V_L$ such that there exists an even length alternating path to v from some unsaturated vertex in V_L. Then

1. every vertex $v \in V_L \backslash S_M$ is saturated (otherwise a path of length 0 exists)
2. every vertex $w \in N(S_M)$ is saturated (otherwise an augmenting path to w exists)
3. no edge (v, w) of M joins $V_L \backslash S_M$ to $N(S_M)$ (otherwise v should belong to S_M)

Thus... $$|M| \geq |V_L \backslash S_M| + |N(S_M)|$$

Matchings in Bipartite Graphs

Taken together with the earlier **Claim** this proves the following:

Theorem[Kőnig 1931]

$$\max_{\text{matchings } M} |M| = \min_{S \subseteq V_L} \{|V_L \setminus S| + |N(S)|\}$$

Matchings in Bipartite Graphs

Corollaries

1. There is an efficient algorithm to construct a maximum matching in a bipartite graph —search for augmenting paths
2. The algorithm is *self-certifying* —the set S_M provides a *certificate* of the optimality of M
3. The set $V_L \backslash S \cup N(S)$ is a *vertex cover* of G. The Theorem establishes the fact that, in bipartite graphs,

$$\max_{\text{matchings } M} |M| = \min_{\text{vertex covers } C} \{|C|\}$$

Today...

Matchings and Network Flows

- network flows
  - definitions
  - relationship with bipartite matchings
  - duality

Coming up...

Reductions and relative hardness of problems

- reductions
  - definitions
  - role(s) in establishing relative hardness
  - examples (review)
- overview of problems with efficient algorithms
  ... and related problems with no known efficient algorithm
- the complexity classes **P** and **NP**
- **NP**-hardness and **NP**-completeness

Network Flows

Definitions

A *capacitated network* is a directed graph G with

1. two distinguished vertices: s (the *source*) and t (the *sink*); and
2. a non-negative number $c(e)$, associated with each edge e, called the *capacity* of e

A *flow* (from s to t) in G is a function $f : E \rightarrow \Re$ that satisfies:

1. (*capacity constraints*) $0 \leq f(e) \leq c(e)$, for all $e \in E$; and
2. (*flow conservation*) $\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$.

The *value* of flow f, denoted $|f|$, is defined as:

$$|f| = \sum_{e \text{ out of } s} f(e)$$

The maximum-flow problem

Given a capacitated network G, find a flow from s to t of maximum value

Network Flows

The maximum-flow problem

As with matchings we can ask:

1. can we bound the value of the maximum flow by some natural property of the network?
2. given a flow, how can we improve (augment) it to increase its value?
3. how will we know when we are done?

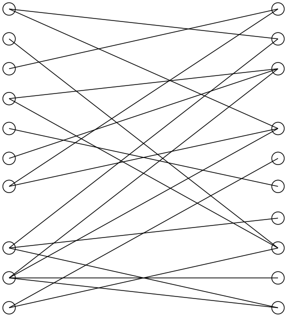
Network Flows and Bipartite Matchings

Maximum bipartite matching can be *reduced* to maximum flow

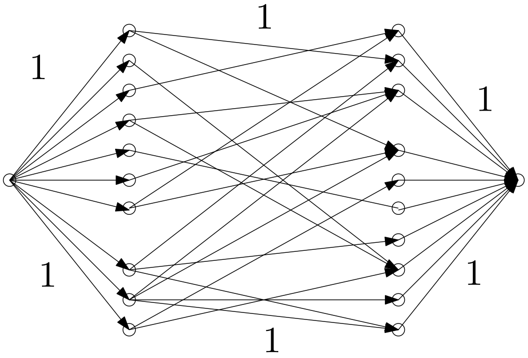
Given a bipartite graph G, with vertex bi-partition V_L and V_R:

1. direct all edges of E from V_L to V_R and assign each one capacity 1
2. create a new source vertex s and sink vertex t
3. add an edge with capacity 1 from s to each vertex in V_L; and
4. add an edge with capacity 1 from each vertex in V_R to t

Network Flows and Bipartite Matchings



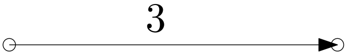
Network Flows and Bipartite Matchings



Network Flows and Bipartite Matchings

Maximum flow in *integer* capacitated networks can be *reduced* to maximum flow in *unit* capacitated networks

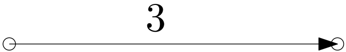
replace



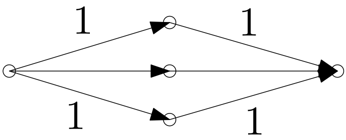
Network Flows and Bipartite Matchings

Maximum flow in *integer* capacitated networks can be *reduced* to maximum flow in *unit* capacitated networks

replace

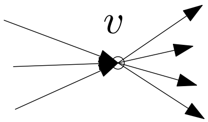


with



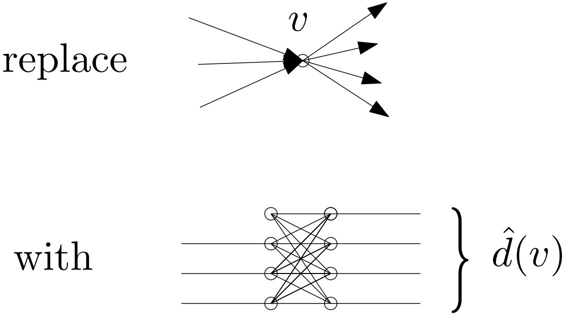
Network Flows and Bipartite Matchings

Maximum flow in *unit* capacitated networks can be *reduced* to maximum bipartite matching

replace 

Network Flows and Bipartite Matchings

Maximum flow in *unit* capacitated networks can be *reduced* to maximum bipartite matching



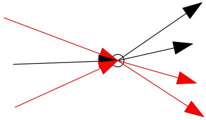
Network Flows and Bipartite Matchings

Claim The resulting bipartite graph has a matching of size $f + \sum_v \hat{d}(v)$ if and only if the original network had a flow of value f.

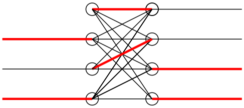
Network Flows and Bipartite Matchings

Matching reflects the flow through a vertex

replace

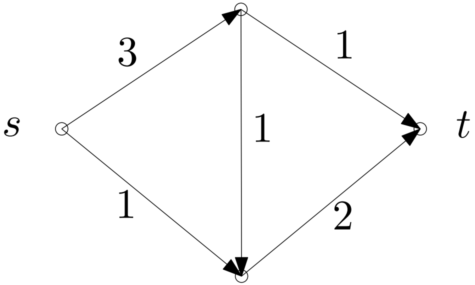


with



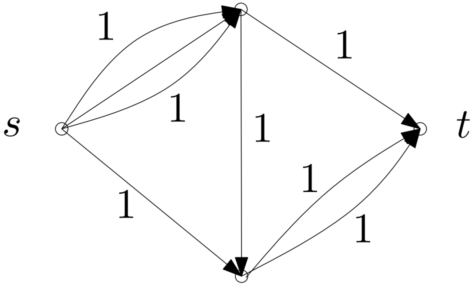
Network Flows and Bipartite Matchings

An example of the full reduction...



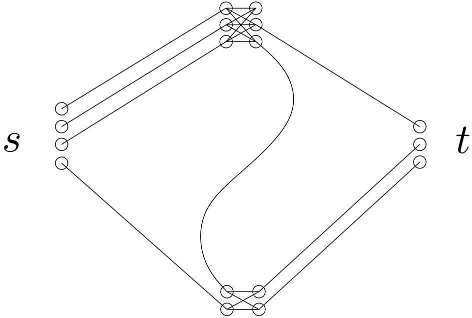
Network Flows and Bipartite Matchings

An example of the full reduction...



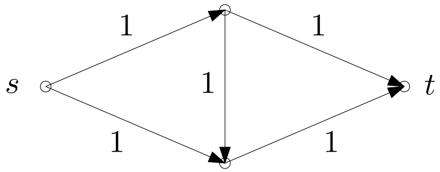
Network Flows and Bipartite Matchings

An example of the full reduction...



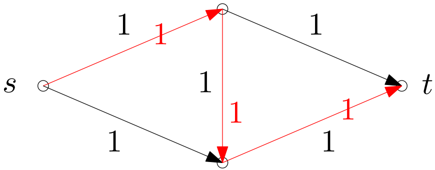
Network Flows and Bipartite Matchings

A capacitated network...



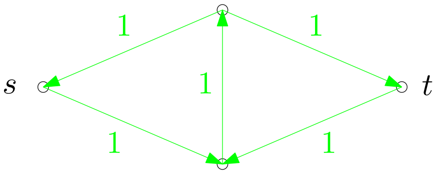
Network Flows and Bipartite Matchings

...and a flow



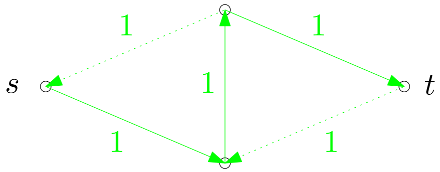
Network Flows and Bipartite Matchings

The residual flow graph...



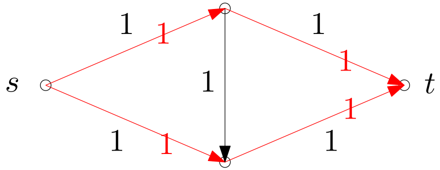
Network Flows and Bipartite Matchings

and associated *augmenting path*



Network Flows and Bipartite Matchings

The augmented flow... (note edge-disjoint paths)



Two Applications...

- resilience of sensor networks
- can the Canucks make the playoffs?

Reductions and relative hardness of problems

We will write A $\leqslant$ B to denote the fact that problem A is *reducible to* problem B. Informally, this means

1. a subroutine for solving problem B can be used as a *black box* in solving problem A
2. instances of problem A can be transformed to instances of problem B in such a way that a solution to the latter can be transformed back into a solution of the former

Reductions and relative hardness of problems

We have seen many examples throughout the course...

- element-distinctness $\lessapprox$ closest-pair $\lessapprox$ sorting
- transitive-closure $\lessapprox$ Boolean-matrix-product
- all-pairs-shortest-paths-with-arbitrary-weights
  $\lessapprox$ all-pairs-shortest-paths-with-non-negative-weights
- edit-distance (sequence-alignment)
  $\lessapprox$ single-source-shortest-path

Reductions and relative hardness of problems

We have seen many examples throughout the course...

- bipartite-matching $\leq$ bipartite-vertex-cover $\leq$ bipartite-matching
- bipartite-matching $\leq$ unit-capacitated-network-flow $\leq$ bipartite-matching
- integer-capacitated-network-flow $\leq$ unit-capacitated-network flow

Reductions and relative hardness of problems

and in homework assignments...

- minimum-colour-transition-path $\leq$ min-cost-shortest-path
- vertex-cover $\leq$ minimum-colour-path

Coming up...

Reductions and relative hardness of problems

- overview of problems with efficient algorithms
  ... and related problems with no known efficient algorithm
- reductions...treated more formally
- the complexity classes **P** and **NP**
- **NP**-hardness and **NP**-completeness

Coming up...

Reductions and relative hardness of problems

- reductions
  - definitions
  - role(s) in establishing relative hardness
  - examples (review)
- overview of problems with efficient algorithms
  ... and related problems with no known efficient algorithm
- the complexity classes **P** and **NP**
- **NP**-hardness and **NP**-completeness