Reductions to Network Flow Problem

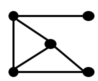
- **Bipartite Matching µ Network Flow**
- The Gee-ball Problem ∝ Network Flow

Matching

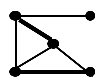
- Let G=(V,E) be a graph.
- E′ ⊆ E is a matching if every vertex of V is incident to at most one edge of E′.

Matching Example

G



E′

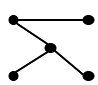


Bipartite Graph

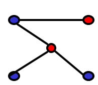
- Let G=(V,E) be a graph.
- G is bipartite if V can be partitioned into V_1 and V_2 such that no pair of vertices in V_i (i=1,2) have an edge.

Bipartite Example

G



G

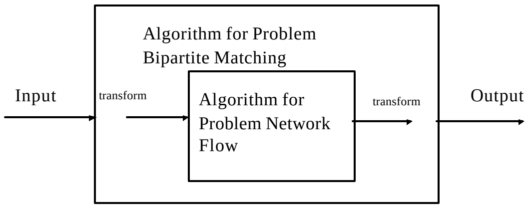


CHECK

Bipartite Matching

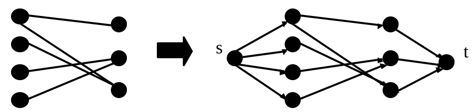
- Input: Bipartite graph G
- Output: A largest matching of G

Bipartite Matching ∝ Network Flow



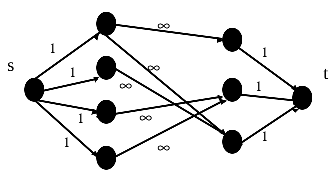
Bipartite Matching ∝ Network Flow

- Transform Input – Step 1

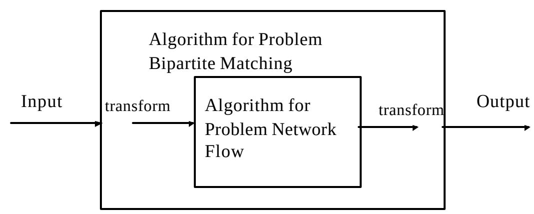


Bipartite Matching ∝ Network Flow

- Transform Input – Step 2



Bipartite Matching ∝ Network Flow



Reductions to Network Flow Problem

- **Bipartite Matching µ Network Flow**
- The Gee-ball Problem ∝ Network Flow

Matching

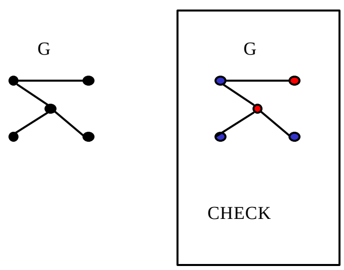
- Let G=(V,E) be a graph.
- E′ ⊆ E is a matching if every vertex of V is incident to at most one edge of E′.

Matching Example

Bipartite Graph

- Let G=(V,E) be a graph.
- G is bipartite if V can be partitioned into V_1 and V_2 such that no pair of vertices in V_i (i=1,2) have an edge.

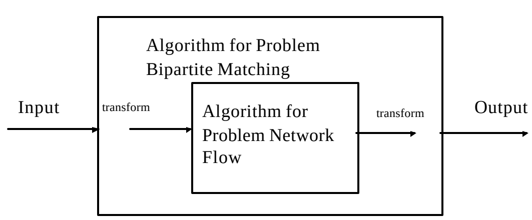
Bipartite Example



Bipartite Matching

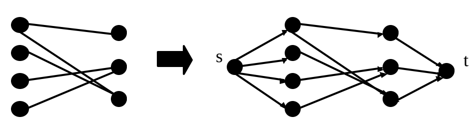
- Input: Bipartite graph G
- Output: A largest matching of G

Bipartite Matching ∝ Network Flow



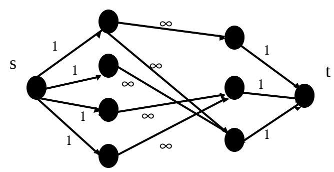
Bipartite Matching ∝ Network Flow

- Transform Input– Step 1

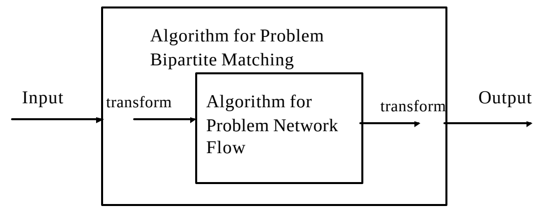


Bipartite Matching ∝ Network Flow

- Transform Input – Step 2

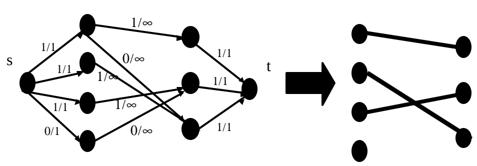


Bipartite Matching ∝ Network Flow



Bipartite Matching ∝ Network Flow

- Transform output



Reduction

- Is it correct?
- Is it efficient?

Integrality theorem

- If the capacities in a network are integral, then the max flow can be achieved with integral flows on each edge.
- Further the Ford-Fulkerson method yields an integral solution.

Proof of correctness

There is a 1-1 correspondence between 0/1 flows in the network and matchings in the input graph.

Reduction

- Is it correct?
- **Is it efficient?**

  $T_{BM}(n)=cn+T_{FF}(n+2)$

Reductions to Network Flow Problem

- Bipartite Matching $\propto$ Network Flow
- **The Gee-ball Problem µ Network Flow**

The Gee-ball Problem

- The southwestern conference of the gee-ball league consists of n+1 teams. Team n+1 is from HMC.
- We want to know whether it is possible for HMC to win more games this season than any other team in the conference.
- No ties allowed.

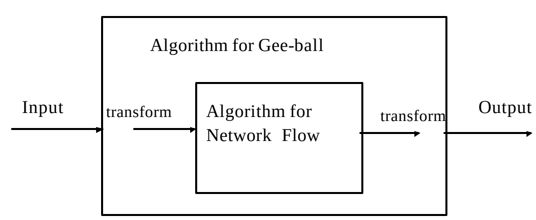
Example

- The teams are Pitzer, CMC, Pomona, and HMC
- Games won so far:
  - Pitzer 4, CMC 3, Pomona 2, HMC 2
- Games to play:
  - 1 game: Pitzer vs. HMC
  - 2 games: Pomona vs. HMC

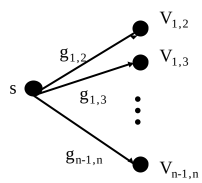
The Gee-ball Problem

- Teams $t_1, t_2, \dots, t_n, t_{n+1}$
- So far this year team i has won w_i games.
- Teams i and j will play each other $g_{i,j}$ more times this season ($g_{i,j}=g_{j,i}$).

Gee-ball $\propto$ Network Flow

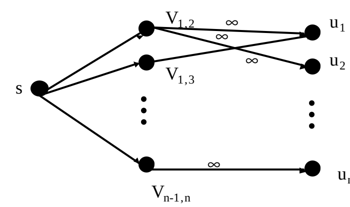


Transform Input



1. Create a source s.
2. Create vertex $v_{i,j}$ for $1 \le i < j \le n$
3. Create edge from s to $v_{i,j}$ with capacity $g_{i,j}$

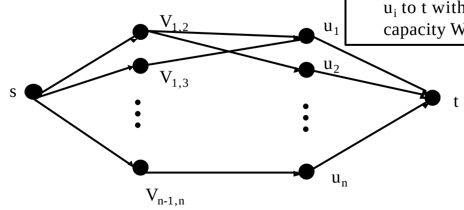
Transform Input



4. Create vertices u_i, $1 \le i \le n$
5. Create edge from $v_{i,j}$ to u_i and u_j with infinite capacity.

Transform Input

5. Create a sink t.
6. Create edge from u_i to t with capacity W_i.



Transform Input

Let $W = w_{n+1} + \Sigma_{i=1\dots n}\, g_{n+1,i}$