

Reduction

(Based on [Manber 1989])

Yih-Kuen Tsay

Department of Information Management
National Taiwan University

Introduction

- 🌐 The basic idea of *reduction* is to solve a problem with the solution to another “similar” problem.
- 🌐 When Problem A can be reduced to Problem B , there are two consequences:
 - ☀️ A solution to Problem B may be used to solve Problem A .
 - ☀️ If A is known to be “hard”, then B is also necessarily “hard”.
- 🌐 One should avoid the pitfall of reducing a problem to another that is too general or too hard.

Matching

- Given an undirected graph $G = (V, E)$, a **matching** is a set of edges that do not share a common vertex.
- A **maximum** matching is one with the maximum number of edges.
- A **maximal** matching is one that cannot be extended by adding any other edge.

Bipartite Matching

- 🌐 A bipartite graph $G = (V, E, U)$ is a graph with $V \cup U$ as the set of vertices and E as the set of edges such that
- ☀️ V and U are disjoint and
 - ☀️ The edges in E connect vertices from V to vertices in U .

Problem

Given a bipartite graph $G = (V, E, U)$, find a maximum matching in G .

- Consider a directed graph, or network, $G = (V, E)$ with two distinguished vertices: s (the **source**) with indegree 0 and t (the **sink**) with outdegree 0.
- Each edge e in E has an associated positive weight $c(e)$, called the *capacity* of e .

The Network Flow Problem

🌐 A **flow** is a function f on E that satisfies the following two conditions:

1. $0 \leq f(e) \leq c(e)$.
2. $\sum_u f(u, v) = \sum_w f(v, w)$, for all $v \in V - \{s, t\}$.

🌐 The **network flow problem** is to maximize the flow f for a given network G .

Bipartite Matching to Network Flow

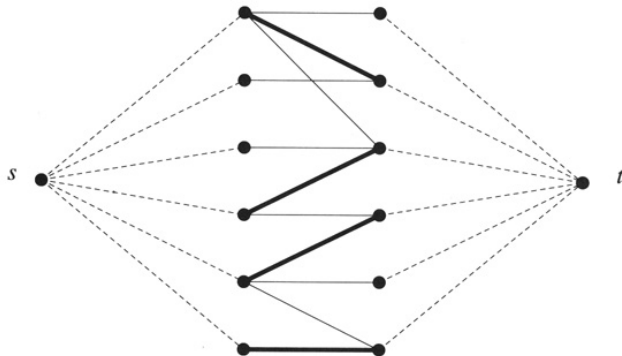


Figure 7.39 Reducing bipartite matching to network flow (the directions of all the edges are from left to right).

Source: [Manber 1989].

🌐 Mapping from the input $G = (V, E, U)$ of the bipartite matching problem to the input $G' = (V', E')$ and c of the network flow problem:

☀️ The network is $G' = (V', E')$ where

👤 $V' = \{s\} \cup V \cup U \cup \{t\}$

👤 $E' = \{(s, v) \mid v \in V\} \cup E \cup \{(u, t) \mid u \in U\}$

☀️ The capacity for every $e \in E'$ is 1, i.e., $\forall e \in E', c(e) = 1$.

🌐 Correspondence between the two solutions

☀️ A maximum flow f in G' defines a maximum matching M_f in G .

☀️ A maximum matching M in G induces a maximum flow f_M in G' .

Notations

- Let \bar{v} denote a vector (v_1, v_2, \dots, v_n) of n constants or n variables.
- In the following, \bar{a} , \bar{b} , \bar{c} , and \bar{e} are vectors of n constants.
- And, \bar{x} and \bar{y} are vectors of n variables.
- The (inner or dot) product $\bar{a} \cdot \bar{x}$ of two vectors \bar{a} and \bar{x} is defined as follows:

$$\bar{a} \cdot \bar{x} = \sum_{i=1}^n a_i \cdot x_i$$

Linear Programming

- Objective function:

$$\bar{c} \cdot \bar{x}$$

- Equality constraints:

$$\begin{bmatrix} \bar{e}_1 \\ \bar{e}_2 \\ \vdots \\ \bar{e}_m \end{bmatrix} \bar{x} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_m \end{bmatrix}$$

- Inequality constraints may be turned into equality constraints by introducing *slack* variables.
- The goal is to *maximize* (or *minimize*) the value of the objective function, subject to the equality constraints.

Network Flow to Linear Programming

🌐 Mapping from the input $G = (V, E)$ and c of the network flow problem to the objective function and constraints of linear programming:

- ☀ Let x_1, x_2, \dots, x_n represent the flow of the n edges.
- ☀ Objective function

$$\sum_{i \in S} x_i$$

where S is the set of edges leaving the source.

- ☀ Inequality constraints

$$x_i \leq c_i, \text{ for all } i, 1 \leq i \leq n$$

where c_i is the capacity of edge i .

- ☀ Equality constraints

$$\sum_{i \text{ leaves } v} x_i - \sum_{j \text{ enters } v} x_j = 0, \text{ for every } v \in V \setminus \{s, t\}$$