Modeling, Analysis and Optimization of Networks

Alberto Ceselli
alberto.ceselli@unimi.it

Università degli Studi di Milano
Dipartimento di Informatica

Doctoral School in Computer Science

A.A. 2015/2016

Summary of Lecture 1

- Networks are pervasive
- Networks are too complex in features and size to be managed without advanced tools
- A suitable formal framework is known to model network flows that
  - is « compact »
  - is flexible
  - is tractable

Summary of Lecture 2
Properties of Flows

- Another good feature of Network Flows :
  - structural properties can be proved by the design of efficient algorithms!
- … i.e. theory and computation fit nicely!

Summary of Lecture 2

- Dijkstra Algorithm with example

  by J. Orlin
- Proof of correctness (by invariants)
- Summary: we have an efficient way for finding (special) paths between nodes of networks
- *Max flow algorithms*

Summary of Lecture 3

- Max flow algorithms
- Flows and Cuts
  - Integrality of flows
  - Max Flow – Min Cut: Weak duality
  - Max Flow – Min Cut: Strong duality
  - Stopping conditions and correctness of Ford-Fulkerson
- Modeling with Cuts (Image segmentation)

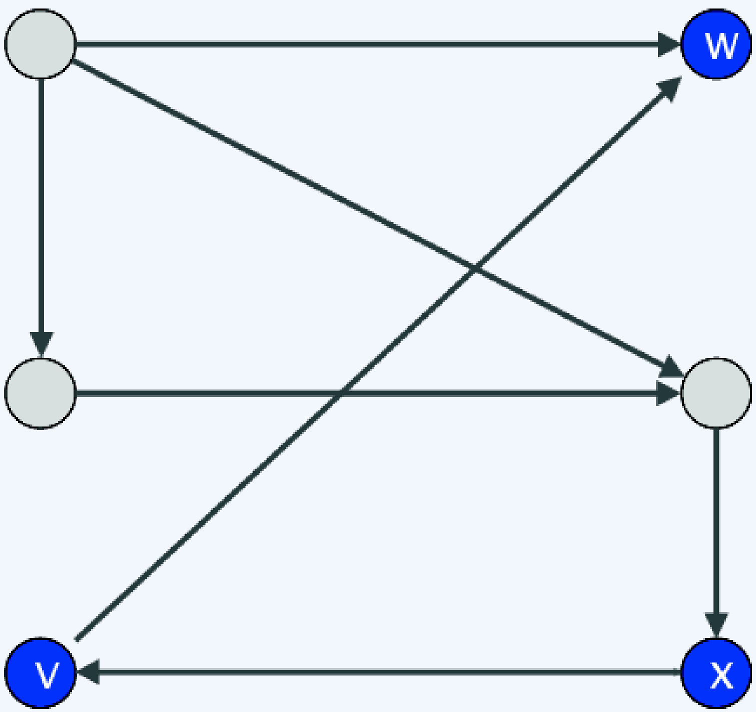
Plan of Lecture 4

- Modeling with Cuts (project selection)
- Paths and Flows
- Minimum cost flow problems
  - Modeling (task scheduling)
  - Properties and algorithms
- Lab session

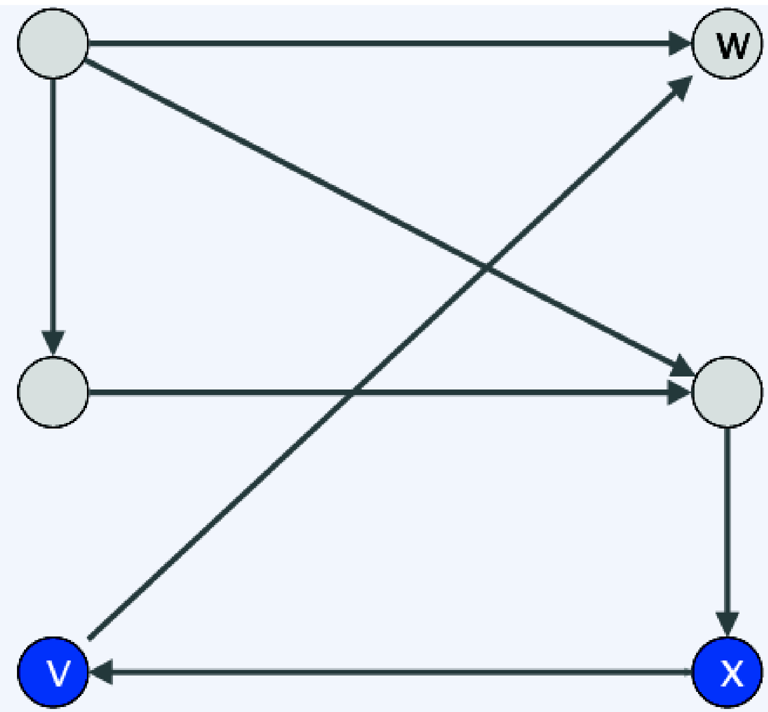
Example 7: project management

Taken from J. Kleinberg, E. Tardos, « Algorithm Design»

- Set of possible projects P
- Each project i has an associated coefficient p_i
  - Profit if $p_i > 0$
  - Cost if $p_i < 0$
- Set of prerequisites E: if (i,j) in E, i cannot be done unless also j is done
- A subset A of projects is feasible if each prerequisite of a project in A is also in A
- Goal: find a feasible subset of projects of maximum revenue (maximum weight closure problem)



{ v, w, x } is feasible



{ v, x } is infeasible

Paths and Flows

- Recall Ford-Fulkerson!
  - We successively send flow across paths
  - Until some terminating conditions are met
- **Flow Decomposition** Theorem
  - Every path (and cycle) flow has a unique representation as nonnegative arc flows.
  - Conversely, every nonnegative arc flow can be represented as a path (and cycle) flow.

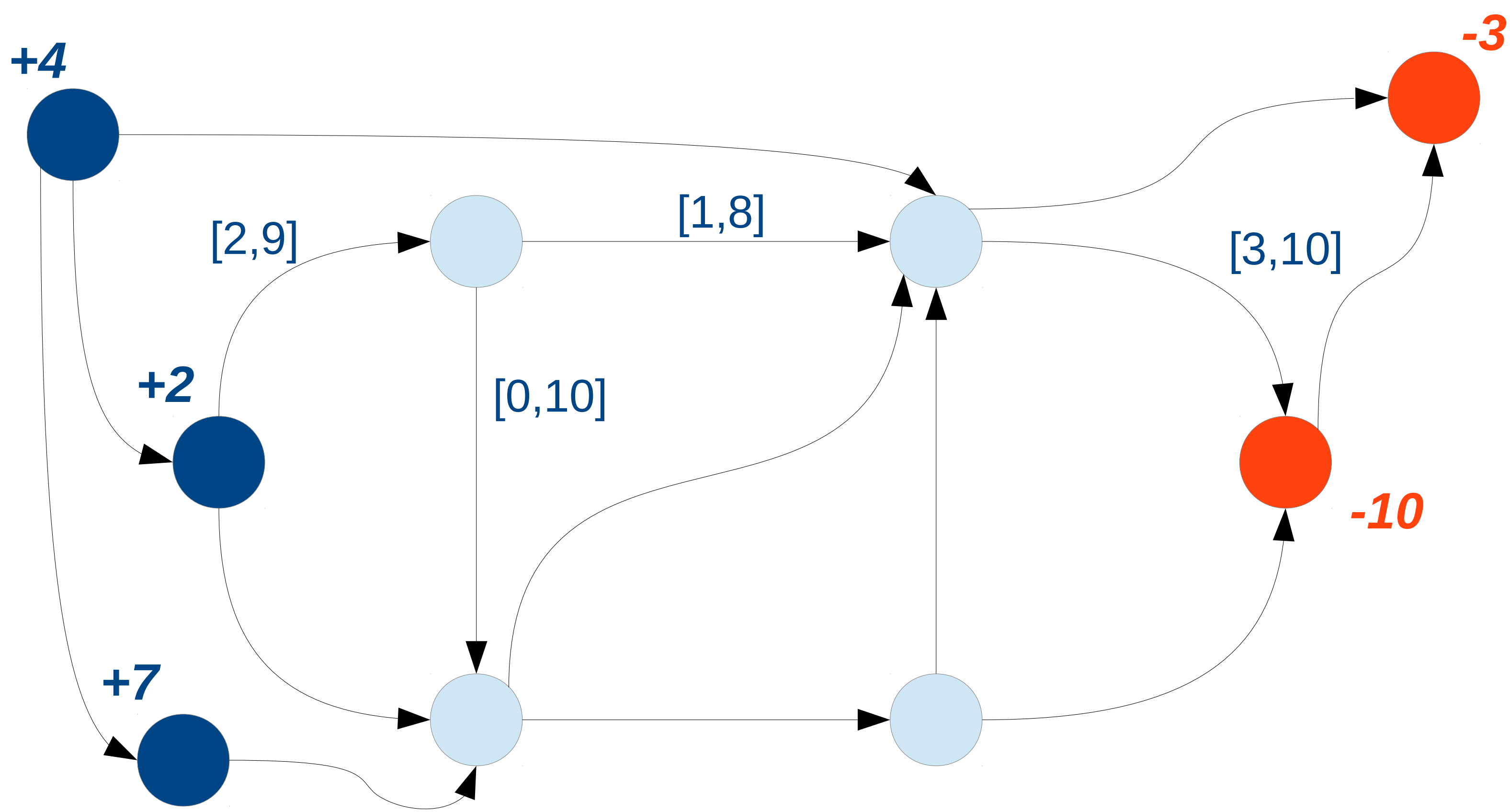
    (a) Every directed path with positive flow connects a deficit node to an excess node

    (b) An most n+m paths (and cycles) have nonzero flow (out of these, at most m cycles have nonzero flow
  - Flow ↔ Path mapping may not be unique
- Proof: constructive! (on the blackboad)

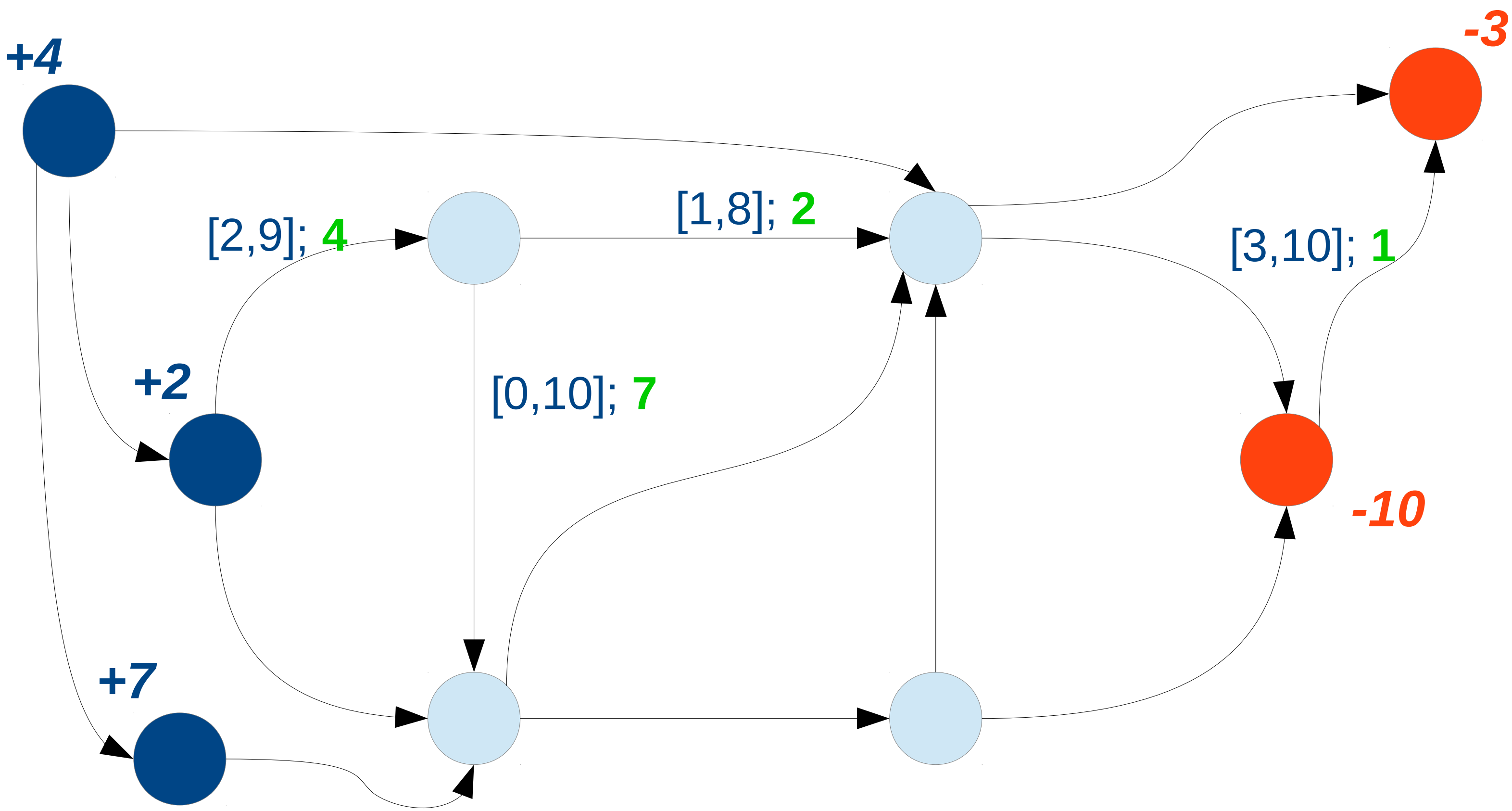
Minimum Cost Flows

- Shortest Path problems
  - Costs on arcs, No capacities
- Max Flow problems
  - Capacities, no costs on arcs
- Let's extend our framework, combining these two features !

Min Cost Flow Problems



Min Cost Flow Problems



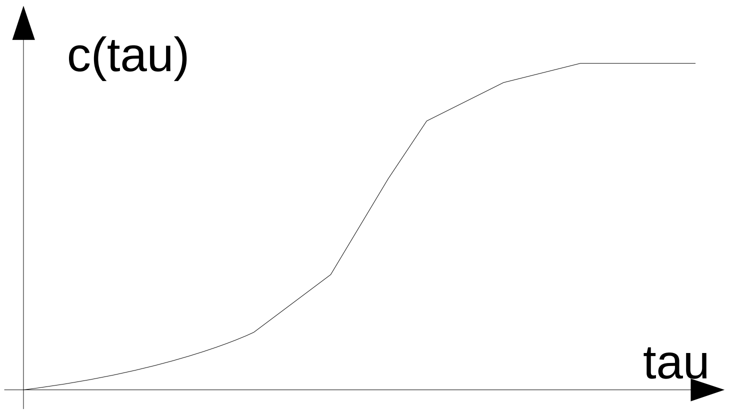
Example 8: Distribution Problems

Network Flows, 9.1

- Car manufacturers need to plan their logistics
- Several plants, each with a maximum production **capacity** (and also a minimum one)
- Several car models
- Several retailers, each with a model **demand**
- Car production **costs**, possibly different on different plants
- Transportation **costs** between plants and retailers
- How to decide where to produce and how to distribute cars?

Example 9: Scheduling with deferral costs

- We have p tasks and q processors
- All tasks have the same processing time t
- Each task has a different deferral cost function c(tau), where tau is the task completion time



- Each processor can perform one task at a time
- How to assign tasks to processors in order to minimize the total deferral cost?

Minimum Cost Flows

- Idea:
  - Keep distance labels d(i), as in shortest path algorithms
  - Work on residual networks, as in maximum flow algorithms
  - Use iterative algorithms « Ford Fulkerson » style

Minimum Cost Flows

- Main theoretical steps: work with **reduced costs** instead of actual costs

  p(i,j) = c(i,j) + d(i) – d(j)

- Theorem: searching for shortest paths (and hence flows) by looking at costs or reduced costs is equivalent
  - Proof (blackboad)
- Theorem: Optimality condition is

  p(i,j) ≥ 0 for each (i,j)
  - Proof omitted

Minimum Cost Flows

- Successive Shortest Path Algorithm:
  - Start with flow = 0 and d(i) = 0 for each node
  - This is a pseudo-flow (only capacities are respected)
  - Iteratively
    - Consider a residual network
    - Consider reduced costs
    - Find a path from an excess to a defect node on the residual network using only arcs of 0 reduced cost
  - Until the pseudo-flow becomes feasible!

Minimum Cost Flows

- Example: Orlin slides

A short lab session

- Using GLPK to model and solve (numerically) Max Flow – Min Cut Probs

Example graph