Introduction to Network Optimization

AU4606: Network Optimization

AI4702: Network Intelligence and Optimization

Xiaoming Duan
Department of Automation
Shanghai Jiao Tong University

September 11, 2023

Welcome

- Welcome to AU4606/AI4702
  - AU4606 (48): Network Optimization
  - AI4702 (32): Network Intelligence and Optimization
- Instructors
  - Xiaoming Duan (44): network flows and optimization
  - Jianping He (2): introduction to multi-agent systems
  - Chongrong Fang (2): introduction to cloud networks
- Main references
  - Ravindra K. Ahuja, Thomas L. Magnanti, James B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, 1993.
  - David P. Williamson, *Network Flow Algorithms*, Cambridge University Press, 2019.
  - Mokhtar S. Bazaraa , John J. Jarvis, Hanif D. Sherali, *Linear Programming and Network Flows*, Wiley, 2009.

Course content

Week 1-8 (AU4606 & AI4702):

- Introduction (this lecture)
- Preparations (3 lectures)
  - basics of graph theory
  - algorithm complexity and data structure
  - graph search algorithm
- Shortest path problems (3 lectures)
- Maximum flow problems (5 lectures)
- Minimum cost flow problems (3 lectures)
- Introduction to multi-agent systems (1 lecture)
- Introduction to cloud networks (1 lecture)

Week 9-16 (AU4606):

- Simplex and network simplex methods (2 lectures)
- Global minimum cut problems (3 lectures)
- Minimum spanning tree problems (3 lectures)

Class times and grading

Class times

- Formulate problems
- Present algorithms (and applications)
- Do proofs (correctness and complexity)

Grading

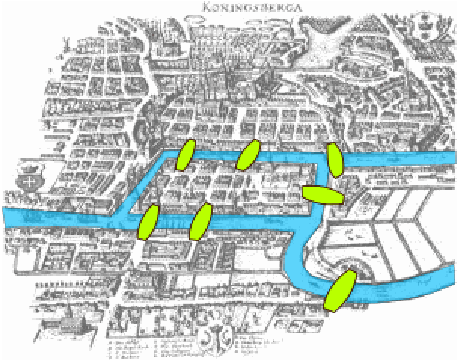
- 20% attendance (which I will be checking VERY occasionally)
- 80% homework on a biweekly basis (might involve coding)
  - 4 problem sets for AI4702
  - 6 problem sets for AU4606

Questions?

Today

- Seven Bridges of Königsberg
- Euler's circuit theorem and its proof
- Minimum cost flow problems and other related problems

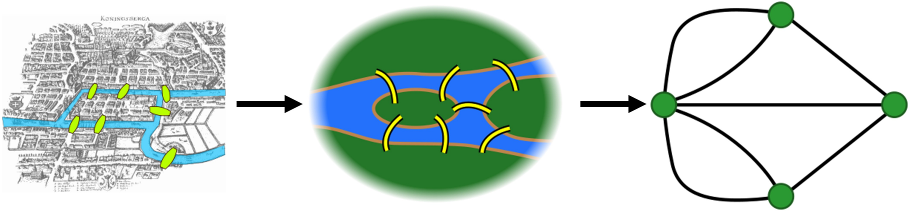
- "Graph theory" began in 1736
- Leonhard Euler visited the city of Königsberg in Prussia



- People wondered whether it is possible to take a walk, end up where you started from, and cross each bridge in Königsberg exactly once

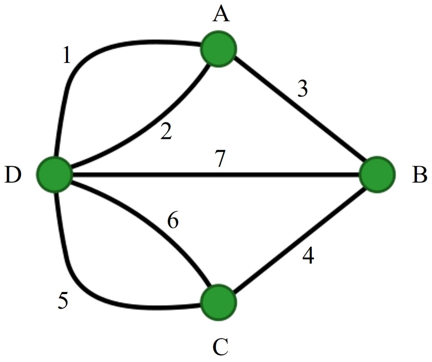
Euler's abstraction

- Landmass are nodes
- Bridges are edges



Seven bridges of Königsberg

Problem transformation

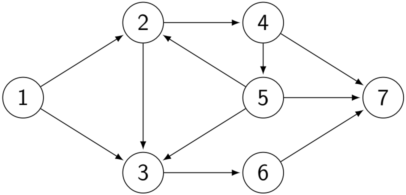


Is there a "walk" starting at A and ending at A and passing through each arc exactly once?

Notation and terminology: graphs

Notation and terminology: graphs

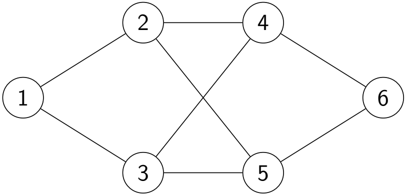
- A graph/network is a pair $G = (N, A)$
  - N: a set of nodes/vertices
  - A: a set of arcs/edges



A directed graph

$$N = \{1, 2, 3, 4, 5, 6, 7\}$$

$$A = \{(1,2), (1,3), (2,3), (2,4), (3,6), (4,7), (5,2), (5,3), (5,7), (6,7)\}$$



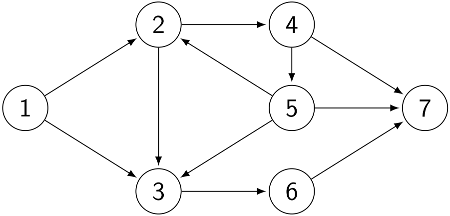
An undirected graph

$$N = \{1, 2, 3, 4, 5, 6\}$$

$$A = \{\{1,2\}, \{1,3\}, \{2,4\}, \{2,5\}, \{3,4\}, \{3,5\}, \{4,6\}, \{5,6\}\}$$

Notation and terminology: paths

In a directed graph $G = (N, A)$

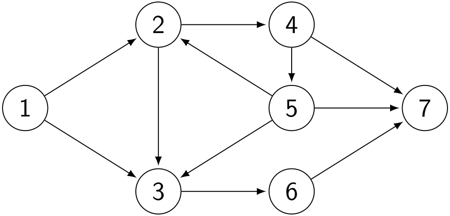


- A path is a sequence of nodes $(i_1, i_2, \ldots, i_r)$ such that
  1. $i_k \neq i_{k'}$ for all $k, k' \in \{1, \ldots, r\}$ and $k \neq k'$, i.e., no node repetition
  2. $(i_k, i_{k+1}) \in A$ or $(i_{k+1}, i_k) \in A$, i.e., directions are ignored

  examples: $(1, 2, 3, 5)$, $(7, 5, 4, 2)$, ~~$(1, 2, 3, 5, 2)$~~
- A directed path is a sequence of nodes $(i_1, i_2, \ldots, i_r)$ such that
  1. $i_k \neq i_{k'}$ for all $k, k' \in \{1, \ldots, r\}$ and $k \neq k'$, i.e., no node repetition
  2. $(i_k, i_{k+1}) \in A$, i.e., directions are respected

  examples: $(1, 2, 3, 6, 7)$, $(4, 5, 7)$, ~~$(2, 5, 3, 1)$~~

In a directed graph $G = (N, A)$



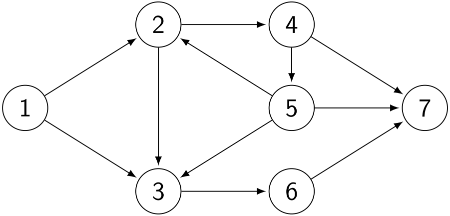
- A cycle is a path $(i_1, i_2, \ldots, i_r)$ such that $i_1 = i_r$
  examples: $(1, 2, 3, 1)$, $(2, 4, 5, 2)$
- A directed cycle is a directed path $(i_1, i_2, \ldots, i_r)$ such that $i_1 = i_r$
  examples: $(2, 4, 5, 2)$

In undirected graphs

- path = directed path
- cycle = directed cycle

Notation and terminology: walks

In a directed graph $G = (N, A)$



- A walk is a sequence of nodes $(i_1, i_2, \dots, i_r)$ such that
  1. $(i_k, i_{k+1}) \in A$ or $(i_{k+1}, i_k) \in A$, i.e., directions are ignored

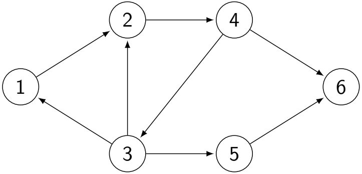
  examples: $(1, 2, 3, 1, 2, 5)$
- A directed walk is a sequence of nodes $(i_1, i_2, \dots, i_r)$ such that
  1. $(i_k, i_{k+1}) \in A$, i.e., directions are respected

  examples: $(1, 2, 4, 5, 2, 3, 6)$
- A walk $(i_1, i_2, \dots, i_r)$ is closed if $i_1 = i_r$

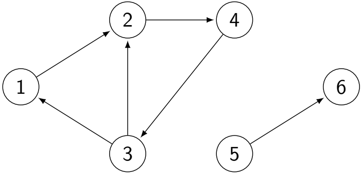
Walks allow node repetition, paths do not

Notation and terminology: connectivity

In a directed graph $G = (N, A)$

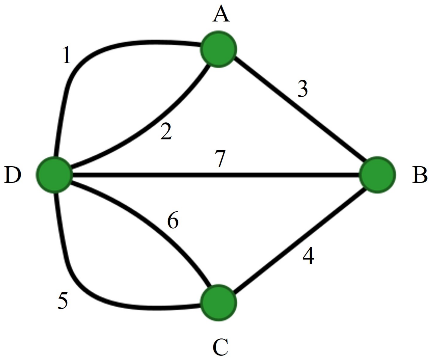


(a) Connected



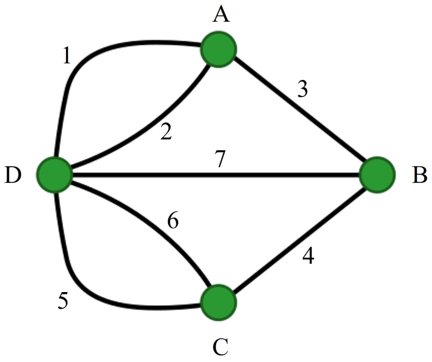
(b) Disconnected

- Nodes i and j are connected if there is at least a path (i.e., ignore directions) from i to j
- A graph is connected if every pair of nodes is connected, otherwise, disconnected



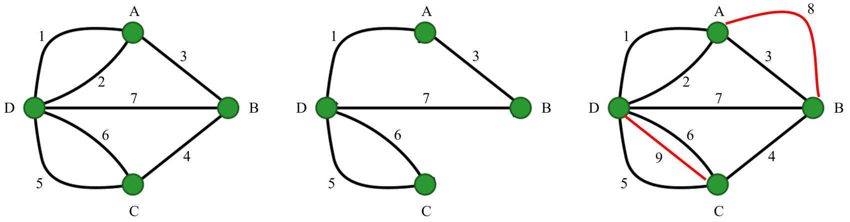
Is there a "walk" starting at A and ending at A and passing through each arc exactly once? – Such a walk is called an Eulerian circuit.

Seven bridges of Königsberg



Is there a "walk" starting at A and ending at A and passing through each arc exactly once? – Such a walk is called an Eulerian circuit.

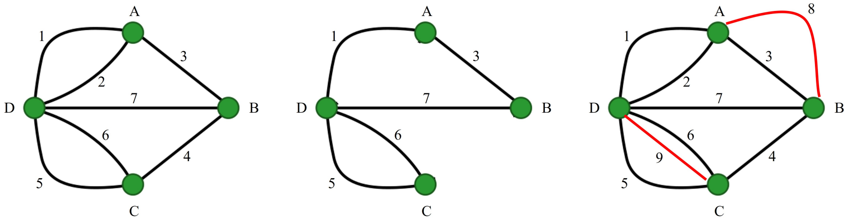
Euler: NO!



(a) No Eulerian circuit (b) Two bridges removed (c) Two bridges added

In (b), an Eulerian circuit: (A,1,D,5,C,6,D,7,B,3,A)

In (c), an Eulerian circuit: (A,1,D,5,C,9,D,6,C,4,B,7,D,2,A,3,B,8,A)



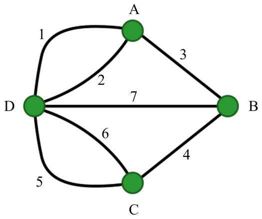
(a) No Eulerian circuit (b) Two bridges removed (c) Two bridges added

In (b), an Eulerian circuit: (A,1,D,5,C,6,D,7,B,3,A)

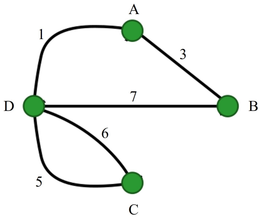
In (c), an Eulerian circuit: (A,1,D,5,C,9,D,6,C,4,B,7,D,2,A,3,B,8,A)

- Degree of a node: the number of incident arcs
    - In (a), deg(A) $= 3$
    - In (b), deg(A) $= 2$
    - In (c), deg(A) $= 4$

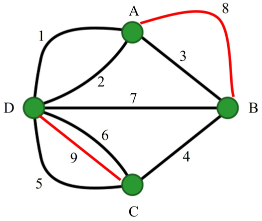
Seven bridges of Königsberg



(a) No Eulerian circuit



(b) Two bridges removed



(c) Two bridges added

Euler's Circuit Theorem

An undirected connected graph has an Eulerian circuit if and only if the degree of every node is even.

The proof does not suggest an "algorithm" to find an Eulerian circuit.

Mental break

Something was wrong

Mental break

Something was wrong

- The abstraction is not a "graph"

Mental break

Something was wrong

- The abstraction is not a "graph"

About Königsberg

- When Euler visited Königsberg in 1735, Königsberg was part of Prussia (German)
- After the World War II, it became part of the Soviet Union
- It is now named Kaliningrad

Mental break

Something was wrong

- The abstraction is not a "graph"

About Königsberg

- When Euler visited Königsberg in 1735, Königsberg was part of Prussia (German)
- After the World War II, it became part of the Soviet Union
- It is now named Kaliningrad

Euler did not draw a graph

Mental break

Something was wrong

- The abstraction is not a “graph”

About Königsberg

- When Euler visited Königsberg in 1735, Königsberg was part of Prussia (German)
- After the World War II, it became part of the Soviet Union
- It is now named Kaliningrad

Euler did not draw a graph
Instead

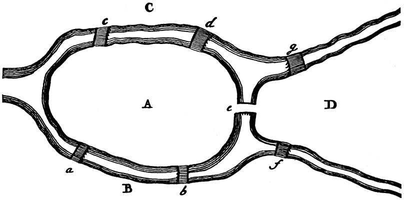
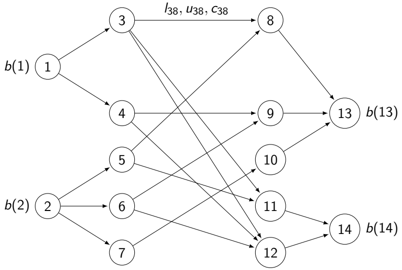
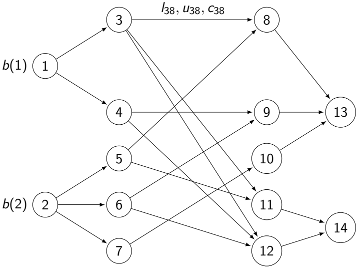


Diagram from Euler’s 1736 paper



- Nodes
  - 1 and 2 are plants, each with a supply $b(i) \geq 0$ for $i \in \{1, 2\}$
  - 13 and 14 are retailers, each with a demand $b(i) \leq 0$ for $i \in \{13, 14\}$
  - intermediate nodes have zero demands, i.e., $b(i) = 0$ for other nodes
  - supply-demand balance $\sum_{i \in N} b(i) = 0$
- Arcs
  - each arc (i, j) has a capacity constraint $[l_{ij}, u_{ij}]$
  - each arc (i, j) has a shipping cost c_{ij}

Send supplies from plants to retailers with minimum cost



Let $G = (N, A)$ be a directed graph, define a variable f_{ij} for each arc

- Objective function

$$\sum_{(i,j)\in A} c_{ij} f_{ij}$$

- Flow balance

$$\sum_{(i,j)\in A} f_{ij} - \sum_{(j,i)\in A} f_{ji} = b(i), \quad \text{for all } i \in N$$

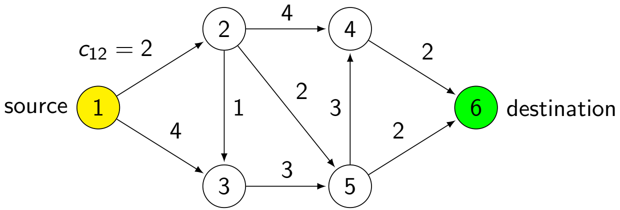
- Flow constraints: $l_{ij} \leq f_{ij} \leq u_{ij}$ for all $(i, j) \in A$

Minimum cost flow: comments

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in A} c_{ij} f_{ij} \\
\text{subject to} \quad & \sum_{(i,j)\in A} f_{ij} - \sum_{(j,i)\in A} f_{ji} = b(i), \quad \text{for all } i \in N \\
& l_{ij} \le f_{ij} \le u_{ij}.
\end{aligned}
$$

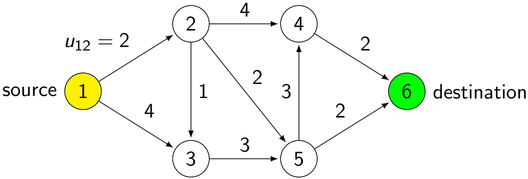
- It is a linear programming
- It is general enough to encompass many other important problems
  - Shortest path problem
  - Maximum flow problem
  - Assignment problem

The shortest path problem



A unit of supply needs to be transported to the destination, what is the shortest directed path?

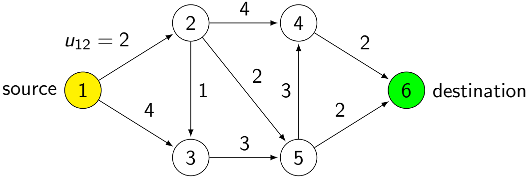
- Set flow bounds $l_{ij} = 0,\ u_{ij} = 1$
- Set supply and demand $b(1) = 1,\ b(6) = -1$, and $b(i) = 0$ for others



What is the maximum supplies that can be sent from source to destination in this network regardless of costs?

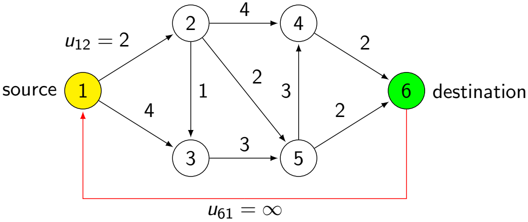
$$
\begin{aligned}
\text{maximize} \quad & b \\
\text{subject to} \quad & \sum_{(i,j)\in A} f_{ij} - \sum_{(j,i)\in A} f_{ji} = 0, \quad \text{for all } i \in N \setminus \{1, 6\} \\
& \sum_{(1,j)\in A} f_{1j} = b, \\
& \sum_{(j,6)\in A} f_{j6} = b, \\
& l_{ij} \leq f_{ij} \leq u_{ij}.
\end{aligned}
$$

The maximum flow problem II

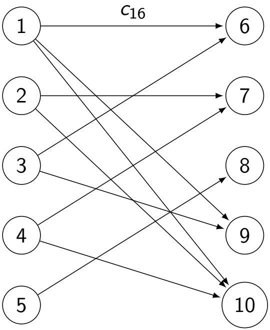


What is the maximum supplies that can be sent from source to destination in this network regardless of costs?

- Set supply and demand $b(i) = 0$ for all nodes
- Set costs $c_{ij} = 0$
- Add an arc $(6, 1)$ with $c_{61} = -1$, $u_{61} = \infty$, add a variable f_{61}



The assignment problem



Assign a set N_1 of workers to a set N_2 of jobs

- Each assignment from $i \in N_1$ to $j \in N_2$ has cost c_{ij}
- Set flow bounds $l_{ij} = 0$, $u_{ij} = 1$
- Set $b(i) = 1$ for $i \in N_1$, and $b(i) = -1$ for $i \in N_2$

Mental break

Our main reference:

Ravindra K. Ahuja, Thomas L. Magnanti, James B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, 1993.

- The relevant topics were developed starting around 1950s

Mental break

Our main reference:
Ravindra K. Ahuja, Thomas L. Magnanti, James B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, 1993.

- The relevant topics were developed starting around 1950s
- There is still ongoing research!

NETWORKS
An International Journal

SPECIAL ISSUE ARTICLE

A fast maximum flow algorithm

James B. Orlin, Xiao-yue Gong

First published 26 November 2020 |
https://doi.org/10.1002/net.22001 | Citations: 2

Funding information: Office of Naval Research, N00014-17-1-2194

Maximum Flow and Minimum-Cost Flow in Almost-Linear Time

Publisher: IEEE Cite This PDF

Li Chen ; Rasmus Kyng ; Yang P. Liu ; Richard Peng ; Maximilian Probst Gutenberg ; Sushant Sachdeva All Authors

Abstract
Document Sections
I. Introduction
II. Overview
Authors
Figures
References

Abstract:
We give an algorithm that computes exact maximum flows and minimum-cost flows on directed graphs with m edges and polynomially bounded integral demands, costs, and capacities in $m^{1+o(1)}$ time. Our algorithm builds the flow through a sequence of $m^{1+o(1)}$ approximate undirected minimum-ratio cycles, each of which is computed and processed in amortized $m^{o(1)}$ time using a new dynamic graph data structure. Our framework extends to algorithms running in $m^{1+o(1)}$ time for computing flows that minimize general edge-separable convex functions to high accuracy. This gives almost-linear time algorithms for several problems including entropy-regularized optimal transport, matrix scaling, p-norm flows, and p-norm isotonic regression on arbitrary directed acyclic graphs.

Published in: 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)

Summary of this lecture

- Seven Bridges of Königsberg
  - Euler's circuit theorem and its proof
- Problem we study in this course: minimum cost flow problems
  - Shortest path problem
  - Maximum flow problem
  - Assignment problem

Upcoming

Week 1-8 (AU4606 & AI4702):

- Introduction (this lecture)
- Preparations (3 lectures)
  - basics of graph theory (next lecture)
  - algorithm complexity and data structure
  - graph search algorithm
- Shortest path problems (3 lectures)
- Maximum flow problems (5 lectures)
- Minimum cost flow problems (3 lectures)
- Introduction to multi-agent systems (1 lecture)
- Introduction to cloud networks (1 lecture)

Week 9-16 (AU4606):

- Simplex and network simplex methods (2 lectures)
- Global minimum cut problems (3 lectures)
- Minimum spanning tree problems (3 lectures)