Draft of the lecture

Piotr Chołda

October 5, 2017

1 Introduction to the course

1.1 Teacher

- Lecture and exam:
  - ⋆ Piotr Chołda, Ph.D., Dr.habil.
  - ⋆ Room: 113, pav. D5 (first floor).
  - ⋆ Office hours: Thursdays, 2.45-4.15PM, and after setting an appointment via e-mail.
  - ⋆ Phone: (617–)26–16.
  - ⋆ E-mail: piotr.cholda@agh.edu.pl.
  - ⋆ WWW: course webpage.
- Laboratory classes and projects:
  - ⋆ Andrzej Kamisiński, M.Sc.
  - ⋆ E-mail: andrzejk@agh.edu.pl.

1.2 Course drafts

- Handouts of the slides are not passed to the students.
- Instead, the lecture drafts are available via the course webpage. They are provided a day before a lecture at the latest.
- When the drafts are changed after the lecture (e.g., correction of typos), please pay attention to the newest version, it is indicated with word **modification** in the course webpage.
- Attendees are supposed to print the drafts and bring them to the lecture.
- After a lecture, a short summary on the lecture content is typically made available via the couse webpage.

1.3 Goals of the course

The detailed goals are given in the syllabus of the course.

Practically, the following skills with be practiced and then checked:

- transformation of a verbally given engineering problem to the optimization tasked based on mathematical programming (formulation skills),
- understanding meaning of a mathematical programming task (interpretation skills),
- implementation of an optimization task to the form useful for processing by the appropriate software (solving skills — practiced mainly during laboratory classes),
- reading of scientific texts and using them for inferring knowledge (creative gaining of knowledge skills — practiced mainly during projects).

Additionally, a part of this course embraces problems of reliability.

1.4 Rules

1. The timetables are available via the course webpage.
2. Components of the course:
   - lectures (not obligatory, but at least two attendants — except for the teacher! — must be present),
   - laboratory classes (obligatory),
   - project (obligatory),
   - exam (obligatory).
3. Credit for the course is based on: credit for the laboratory exercises and the project, and the oral exam.
4. Exam: only for persons who got credit for the laboratory exercises and the project.
5. Final grade — only for persons who passed the exam, the grade is based on the arithmetic mean with the following weights: grade for the laboratory exercises (35%), project (35%), and the exam (30%). The grade is found according to the AGH Bylaw.
6. Laboratory exercises:
   (a) See the rules at the webpage: *Laboratory classes: rules*.
   (b) Components:
      - seven meetings,
      - quick (10-min) quiz tests at the beginning of some classes,
      - a practical test during the last classes,
      - activity during classes.
   (c) Laboratory quiz tests: contents of the lecture.

7. Project:
   (a) See the rules at the webpage: *Project: rules*.
   (b) Deadlines for obligatory meetings.
   (c) 5-page long report.
   (d) Public presentation of the project at the end is necessary.
8. Exam: oral.
9. Topics that can be presented by the attendees and the related schedule are given at Google Docs. The required format: LaTeX, `beamer` class: the template is given at the course webpage,

1.5 Basic bibliography for the course

Generally, the course is based on the following books:

1. Michał Pióro and Deepankar Medhi. *Routing, Flow and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers—Elsevier, San Francisco, CA, 2004: network optimization.
2. Poompat Saengudomlert. *Optimization for Communications and Networks*. CRC Press/Science Publishers, Boca Raton, FL, 2012: optimization of network design.
3. Kishor S. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. John Wiley & Sons, Inc., New York, NY, 2016: reliability modeling.

2 Basics of Graph Theory

2.1 Revision on Graph Theory

1. Graph as an abstraction of a telecommunication or computer network.
2. Mathematical description:
   - $G = (V, E)$ (formally graph is defined as a set, but obviously we use the graphical representation; there is also the other approach, i.e., the algebraic graph theory),
   - $v \in V$: *vertex* (pl. vertices), known as 'node' in engineering,
   - $e \in E$: *edge*, $e = \{v, w\} \in E$, $v, w \in V$, $v \neq w$, known as 'link' in engineering,
   - some vertices are *adjacent*/*neighboring*/*connected*, while some vertices and edges are *incident*.
3. Node *degree* $deg(v)$:
   - $|E| = \frac{1}{2}\sum_{v\in V} deg(v)$,
   - average degree $\mathbb{E}[deg] = \frac{1}{|V|}\sum_{v\in V} deg(v_i) = 2\frac{|E|}{|V|}$,

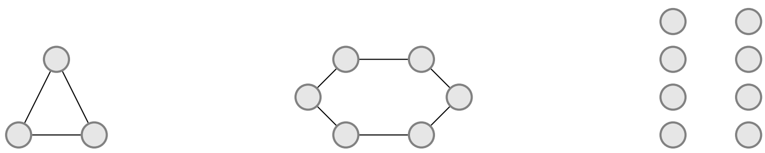
- the handshaking theorem.

4. *Adjacency matrix*: $\mathbf{A} = [a_{ij}]$, $deg(v_i) = \sum_{j=1}^{|V|} a_{ij}$; *incidence matrix*.
5. *Walk* in a graph; relationship between the number of walks and $\mathbf{A}$.
6. *Path* in a graph:
   - $< v, w >$: n-tuple of vertices $< v = v_1, v_2, \ldots, v_{n-1}, v_n = w >$, such that $\{v_i, v_{i+1}\}$ is an edge and the edges are not repeated,
   - path *length*: $n - 1$.
7. *Cycle* in a graph.
8. *Distance* between vertices, $dist(i, j)$ (the shortest path between the vertices).
9. *Diameter* of a graph, $d(G)$ (the longest distance).
10. *Subgraph* of a graph.
11. Graph connectivity:
    - connected graphs (subgraphs),
    - components/clusters,
    - edge partition set, edge cut, a *bridge*,
    - edge *connectivity* $\lambda(G)$,
    - vertex partition set, vertex cut, *articulation point/cut vertex*,
    - vertex connectivity $\kappa(G)$,
    - general relationship: $\kappa(G) \leq \lambda(G) \leq deg_{\min} \leq 2\frac{|E|}{|V|}$,
    - the Menger's theorem.
12. Types of graphs:
    - *full mesh* (*complete* graph) K_n,
    - cyclic C_n,
    - linear P_n,
    - *bipartite* graph $G(V_1 \cup V_2, E)$, full bipartite graph $K_{n,m}$,
    - *star* $K_{1,n-1}$,
    - *hypercube* Q_n.
13. **Trees** T_n in graphs:
    - *pendant/leaf* (pl. leaves),
    - a *forest*,
    - $n - 1$: number of vertices in T_n,
    - a *binary tree*, diameter of a full binary tree.
14. *Spanning tree* in a graph.

15. *Weighted graph*, $G = (V, E, f)$, $f : E \to \mathbb{R}$. Vertex *strength*.
16. Euler's cycle, Eulerian graphs, the Euler's theorem, the bridges of Königsberg.
17. Hamilton's cycle, Hamiltonian graphs.
18. Travelling salesman (salesperson) problem. Chinese postman (route inspection) problem.
19. *Digraph*, *arc*, $G = (V, A)$, $A \subseteq \{(x, y) : x, y \in V, x \neq y\}$.
20. *Network* as a weighted digraph, *capacity* vs. *throughput*, *flow*, *source*, *sink*, *saturated* edge/link.

2.2 Exercises

1. Draw a graph containing exactly six cycles and find its diameter.
2. Show which of the graphs shown in Figure 1 is a subgraph of the Petersen's graph given in Figure 2? Justify your selection by labeling vertices.



(a) (b) (c)

Figure 1: Graphs related to exercise 2.

3. Find edge connectivity of a full mesh graph containing 45 edges.
4. Draw a graph G, for which the minimum vertex degree is k and the following relationships between connectivities hold: $\kappa(G) < \lambda(G) < k$.
5. Is hypercube Q_3 a bipartite graph?
6. Is hypercube Q_4 an Eulerian graph?
7. Is hypercube Q_3 a Hamiltonian graph? If yes, give a method to construct the Hamiltonian cycle in this graph.

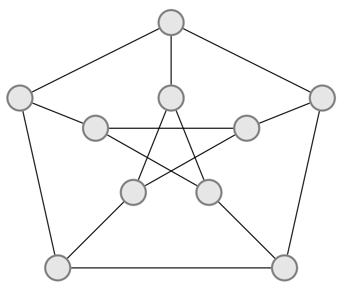


Figure 2: Petersen's graph.

8. What is the difference between the vertex and edge connectivities of a hypercube with containing 32 vertices?

9. Select a natural number n, so that $10 \leq n < 20$. Draw a connected graph having n vertices and not containing cycles. Find density of this graph. Label the vertices with numbers $1, \ldots, n$. Then, modify this graph by adding new edges (and still having a simple graph), so that the adjacency matrix $\mathbf{A}$ can be characterized (after the modification) with the following property: $\forall_{j=1,\ldots,n} \left(\mathbf{A}^2\right)_{jj} = X$ (you can decide value of X on your own).

2.3 Auxiliary reference

- Piet Van Mieghem. *Graph Spectra for Complex Networks.* Cambridge University Press, Cambridge, UK, 2011: algebraic graph theory.