COURSE DESCRIPTION CARD - SYLLABUS

Course name

Combinatorial Optimization

Course

Field of study	Year/Semester
Computing	2/4
Area of study (specialization)	Profile of study
	general academic
Level of study	Course offered in
First-cycle studies	Polish
Form of study	Requirements
part-time	compulsory

Number of hours

Lecture	Laboratory classes	Other (e.g. online)
12		
Tutorials	Projects/seminars	
	12	

Number of credit points

3

Lecturers

Responsible for the course/lecturer:

Maciej Machowiak, Ph.D.

email: Maciej.Machowiak@cs.put.poznan.pl

tel: 616652982

wydział: Computer Science

adres: ul. Piotrowo 2, 60-965 Poznań

Responsible for the course/lecturer:

Prerequisites

A student beginning this subject of study should have basic understanding of discrete mathematics (set theory, logic, graph theory), methods of algorithm design, basic programming structures, abstract data types (e.g. lists, stacks, queues, arbitrary graphs), typical algorithms (e.g. sorting, search in data structures), also basic knowledge on the computational complexity of algorithms and problems.
The student should be able to design basic algorithms and code them, to recognize basic discrete structures, to estimate computational complexity of algorithms, as well as acquire information from the indicated sources.
The student should understand the necessity of expanding his/her competences and be ready to undertake cooperation in a team. As far as social competences are considered, the student must be honest, responsible, persevering, curious, creative, respectful to other people.

Course objective

Course goals:

Introduction into basic problems of combinatorial optimization and the methods of solving them. In particular:

1. acquiring ground understanding on optimizing problems with discrete nature,
2. demonstrating solvability barrier arising from exponential computational complexity of algorithms and computational hardness of problems and stimulate understanding consequences of this barrier,
3. developing a skill of recognizing hard combinatorial optimization problems,
4. familiarizing with the methodology of analyzing and practically solving of computationally hard optimization tasks for problems with discrete nature.

Course-related learning outcomes

Knowledge

ordered and theoretically grounded general knowledge on key issues of computer science, the issues of the current subject (K1st_W4)

knowledge on important directions and developments of computing, and related areaas (K1st_W5)

know basic methods, techniques and tools applied in solving simple cases of analyzing computational complexity of algorithms and discrete problems (K1st_W7)

Skills

design and conduct simple experiments, in particular computer measurements and simulations, analyze obtained results and draw conclusions (K1st_U3)

apply analytical and experimental methods to solve computer science methods (K1st_U4)

estimate computational complexity of algorithms and problems (K1st_U8)

design and code algorithms using at least one popular tool (K1st_U11)

Social competences

understands that knowledge and skills in computer science quickly change and deprecate (K1st_K1)

understands the meaning of knwoledge in solving engineering problems, knows examples engineering problems leading to social losses (K1st_K2)

Methods for verifying learning outcomes and assessment criteria

Learning outcomespresented above are verified as follows:

Formative assessment:

a) lectures:

- based on answers to question asked and open problems posed during the lectures,

b) labs:

- evaluation of the correctness of the programs solving the assigned combinatorial optimization problems
- evaluation of student’s knowledge necessary to prepare, and carry out the lab tasks

Total assessment:

a) lectures:

- based on answers to question in a written test,

b) labs:

- monitoring students activities during classes,
- evaluation of reports on the method and computer program solving the assigned combinatorial optimization problems

Additional elements cover:

- punctuality: additional points for providing solutions (programs) and reports on time
- efficiency (time, quality) of the solutions delivered by the student programs
- ability to work in a team solving a lab assignment
- recommendations improving the teaching process.

Programme content

The lecture covers the following topics: Computational complexity of optimization problems: NP-hardness. The notion of approximation algorithms, examples of approximation algorithms. Hardness of approximation. Computationally easy combinatorial optimization problems: Shortest paths in graphs: Dijkstra's algorithm, DAG algorithm, all-pair shortest paths algorithm. Transitive closure of a binary relation: Floyd-Warshall algorithm. Network flows and related problems: maximum flow problem, Dinic algorithm. flows with minimum arc flow, minimum cost flows, applications of max flow problem in solving scheduling problems. Graph coloring problem: formulation, applications, algorithms. Scheduling problems: formulation, applications, algorithms. Traveling Salesman problem with triangle's inequality.

During the lab-classes students solve NP-hard combinatorial optimization problems. It is required to design and code at least two algorithms solving the assigned problem: a fast method (e.g. greedy algorithm) and of improved quality solutions method (e.g. a branch and bound or metaheuristic method).

Teaching methods

Lecture: multimedia presentation, illustrated with examples given on the board.

Bibliography

Basic

1. Złożoność obliczeniowa problemów kombinatorycznych, J. Błażewicz , WNT, W-wa, 1988

2. Scheduling Computer and Manufacturing Processes, J. Błażewicz, K. Ecker, E.Pesch, G. Schmidt, J. Węglarz , Springer, Berlin, New York, 2001

3. Kombinatoryka dla programistów, W. Lipski , WNT, W-wa, 1982

4. Computers and intractability: A guide to the theory of NP-completeness, M.R.Garey, D.S.Johnson, W.H.Freeman, San Francisco, 1979

5. Algorytmy optymalizacji dyskretnej z programami w języku Pascal, M.Sysło, N.Deo, J.Kowalik, PWN, Warszawa, 1993

6. Wprowadzenie do algorytmów, T.Cormen, C.Leiserson, R.Rivest, C.Stein, WNT, Warszawa, 2005

Additional

1. Badania operacyjne dla informatyków, J.Błażewicz, W.Cellary, R.Słowiński, J.Węglarz, WNT, W-wa, 1983

2. Scheduling malleable tasks on parallel processors to minimize the makespan, J. Błażewicz, M. Machowiak, J. Węglarz, M.Y. Kovalyov, D. Trystram, Annals of Operations Research 129 (1-4), 65-80

Breakdown of average student's workload

	Hours	ECTS
Total workload	72	3
Classes requiring direct contact with the teacher	26	1,0
participating in project classes: 15hours[1]	12	0,5
finalizing project reports (student	12	0,5
coding, running, verifying, and testing performance of the algorithms (student	10	0,4
attending lectures	12	0,5
reading and learning from the indicated literature and other sources (approx. 10 pages per hour), approx.100 pages	16	0,7
learning for the final exam, and writing the test	10	0,4

[1] delete or add other activities as appropriate