# COURSE DESCRIPTION CARD - SYLLABUS

Course name
Combinatorial Optimization [S1Inf1>OKOM]

## Course

Field of study
Computing

Year/Semester
2/3

Area of study (specialization)
–

Profile of study
general academic

Level of study
first-cycle

Course offered in
Polish

Form of study
full-time

Requirements
compulsory

## Number of hours

Lecture
30

Laboratory classes
0

Other (e.g. online)
0

Tutorials
0

Projects/seminars
16

## Number of credit points

4,00

## Coordinators

prof. dr hab. inż. Maciej Drozdowski
maciej.drozdowski@put.poznan.pl

## Lecturers

## 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

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:
Student
1. has ordered and theoretically grounded general knowledge on key issues of computer science, the issues of the current subject
2. knows important directions and developments of computing, and related areas
3. knows basic methods, techniques and tools applied in solving simple cases of analyzing computational complexity of algorithms and discrete problems

Skills:
Student is able to:
1. design and conduct simple experiments, in particular computer measurements and simulations, analyze obtained results and draw conclusions
2. apply analytical and experimental methods to solve computer science methods
3. estimate computational complexity of algorithms and problems
4. design and code algorithms using at least one popular tool

Social competences:
Student understands that:
1. knowledge and skills in computer science quickly change and deprecate
2. the importance of solving engineering problems, knows example engineering problems leading to social losses

## Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

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) project:
- 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 exam,
b) project:
- 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 following points are considered:
1 Complexity theory framework for CO problems analysis

2 Solving CO problems in practice
3 Greedy algorithms and matroids
4 Shortest paths
5 Network flows
6 Matching in bipartite graphs
7 Graph coloring
8 Packing and cutting

## Course topics

The lecture covers the following topics: Pseudopolynomial dynamic programming algorithms for partition and knapsack problems. Strong NP-hardness. Computational complexity of optimization problems: NP-hardness. The notion of approximation algorithms, examples of approximation algorithms. Hardness of approximation. Practically solving hard combinatorial problems. 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, matching in a bipartite graph, applications of max flow problem in solving scheduling problems and graph partitioning. Minimum spanning tree: Kruskal and Prim algorithms. The notion of a matroid. Graph coloring problem: formulation, applications, algorithms. Packing and cutting: formulation, applications, bin packing problem, algorithms for bin packing.

During the project-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.
Project: practical solving combinatorial optimization problems by coding their solutions, conducting computational experiments, discussion on the chosen methods, team work.

## Bibliography

Basic
1. J. Błażewicz, Złożoność obliczeniowa problemów kombinatorycznych, WNT, W-wa, 1988
2. W. Lipski, Kombinatoryka dla programistów, WNT, W-wa, 1982
3. M.R.Garey, D.S.Johnson, Computers and intractability: A guide to the theory of NP-completeness, W.H.Freeman, San Francisco, 1979
4. W.Cook, W.Cunningham, W.Pulleyblank, A.Schrijver, Combinatorial optimization, Wiley & Sons, 1998
5. M.Sysło, N.Deo, J.Kowalik, Algorytmy optymalizacji dyskretnej z programami w języku Pascal, PWN, Warszawa, 1993
6. T.Cormen, C.Leiserson, R.Rivest, C.Stein, Wprowadzenie do algorytmów, WNT, Warszawa, 2005
7. M.Kubale (redaktor), Optymalizacja dyskretna modele i metody kolorowania grafów, WNT, Warszawa, 2003.
Additional
1. J. Błażewicz, K. Ecker, E.Pesch, G. Schmidt, J. Węglarz, Scheduling Computer and Manufacturing Processes, Springer, Berlin, New York, 2001
2. J.Błazewicz, W.Cellary, R.Słowinski, J.Weglarz, Badania operacyjne dla informatyków, WNT, W-wa, 1983
3. L.Banachowski, A.Kreczmar, Elementy analizy algorytmów, WNT, W-wa, 1989;
4. A.V.Aho, J.E.Hopcroft, J.D.Ullman, Projektowanie i analiza algorytmów komputerowych, PWN, W-wa, 1983
5. K.Manuszewski, Grafy Algorytmicznie trudne do kolorowania, praca doktorska, WETI, Gdańsk, 1997
6. M.Drozdowski, D.Kowalski, J.Mizgajski, D.Mokwa, G.Pawlak, Mind the gap: a heuristic study of subway tours, Journal of Heuristics vol.20, Issue 5, October 2014, pp 561-587, DOI 10.1007/s10732-014-9252-3
7. J.Marszałkowski, D.Mokwa, M.Drozdowski, Ł.Rusiecki, H.Narożny, Fast algorithms for online construction of web tag clouds, Engineering Applications of Artificial Intelligence, vol. 64 (2017) pp. 378-

390 DOI: 10.1016/j.engappai.2017.06.023

## Breakdown of average student's workload

| | Hours | ECTS |
|---|---|---|
| Total workload | 100 | 4,00 |
| Classes requiring direct contact with the teacher | 48 | 2,00 |
| Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation) | 52 | 2,00 |