STUDY MODULE DESCRIPTION FORM

Name of the module/subject		Code
Combinatorial Optimization		**1010511331010510332**

Field of study	Profile of study (general academic, practical)	Year /Semester
Computing	**general academic**	**2 / 3**
Elective path/specialty	Subject offered in:	Course (compulsory, elective)
-	**Polish**	**obligatory**

Cycle of study:	Form of study (full-time,part-time)
First-cycle studies	**full-time**

No. of hours				No. of credits
Lecture: **30**	Classes: **-**	Laboratory: **15**	Project/seminars: **-**	**3**

Status of the course in the study program (Basic, major, other)	(university-wide, from another field)
major	**from field**

Education areas and fields of science and art	ECTS distribution (number and %**)**

Responsible for subject / lecturer:

Maciej Drozdowski
email: Maciej.Drozdowski@cs.put.poznan.pl
tel. 616652981
Computer Science
ul. Piotrowo 2, 60-965 Poznań

Prerequisites in terms of knowledge, skills and social competencies:

1	**Knowledge**	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.
2	**Skills**	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.
3	**Social competencies**	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.

Assumptions and objectives of the course:

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.

Study outcomes and reference to the educational results for a field of study

Knowledge:

1. acquire knowledge on algorithms and their complexity - [K1st_W4]

2. have particular knowledge on algorithm design - [K1st_W5]

3. 3. know basic methods, techniques and tools applied in solving simple cases of analyzing computational complexity of algorithms and discrete problems - [K1st_W7]

4. 4. understand consequences of the computational hardness of problems and know practical methods of solving such problems - [-]

Skills:

1. 1. design and conduct simple experiments, in particular computer measurements and simulations, analyze obtained results and draw conclusions - [K1st_U3]

2. apply analytical and experimental methods to solve computer science methods - [K1st_U4]

3. estimate computational complexity of algorithms and problems - [K1st_U8]

4. design and code algorithms using at least one popular tool - [K1st_U11]

Social competencies:

1. understands that knowledge and skills in computer science quickly change and deprecate - [K1st_K1]

2. understands the meaning of knwoledge in solving engineering problems, knows examples engineering problems leading to social losses - [K1st_K2]

Assessment methods of study outcomes

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 exam,

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.

Course description

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. 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 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).

Basic bibliography:

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. Combinatorial optimization, W.Cook, W.Cunningham, W.Pulleyblank, A.Schrijver, Wiley & Sons, 1998
6. Algorytmy optymalizacji dyskretnej z programami w języku Pascal, M.Sysło, N.Deo, J.Kowalik, PWN, Warszawa, 1993
7. Wprowadzenie do algorytmów, T.Cormen, C.Leiserson, R.Rivest, C.Stein, WNT, Warszawa, 2005
8. Optymalizacja dyskretna modele i metody kolorowania grafów, pod red. M.Kubale, WNT, Warszawa, 2003.
9. Złożoność obliczeniowa problemów kombinatorycznych, J. Błażewicz , WNT, W-wa, 1988
10. Scheduling Computer and Manufacturing Processes, J. Błażewicz, K. Ecker, E.Pesch, G. Schmidt, J. Węglarz , Springer, Berlin, New York, 2001
11. Kombinatoryka dla programistów, W. Lipski , WNT, W-wa, 1982
12. Computers and intractability: A guide to the theory of NP-completeness, M.R.Garey, D.S.Johnson, W.H.Freeman, San Francisco, 1979
13. Combinatorial optimization, W.Cook, W.Cunningham, W.Pulleyblank, A.Schrijver, Wiley & Sons, 1998
14. Algorytmy optymalizacji dyskretnej z programami w języku Pascal, M.Sysło, N.Deo, J.Kowalik, PWN, Warszawa, 1993
15. Wprowadzenie do algorytmów, T.Cormen, C.Leiserson, R.Rivest, C.Stein, WNT, Warszawa, 2005
16. Optymalizacja dyskretna modele i metody kolorowania grafów, pod red. M.Kubale, WNT, Warszawa, 2003.

Additional bibliography:

1. 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
2. Jakub 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

Result of average student's workload

Activity	Time (working hours)
1. participating in laboratory classes: 15hours	15
2. finalizing lab reports (student's own work in the off lab hours)	5
3. coding, running, verifying, and testing performance of the algorithms (student's own work in the off lab hours)	10
4. attending lectures	30
5. reading and learning from the indicated literature and other sources (approx. 10 pages per hour), approx.100 pages	10
6. learning for the final exam, and writing the exam	10

Student’s workload

Source of workload	hours	ECTS
Total workload	80	3
Contact hours	45	2
Practical activities	30	2