# COURSE DESCRIPTION CARD - SYLLABUS

Course name
Bio-inspired algorithms and models [S2Inf1-SzInt>AMIB]

## Course

Field of study
Computing

Year/Semester
1/2

Area of study (specialization)
Artificial Intelligence

Profile of study
general academic

Level of study
second-cycle

Course offered in
Polish

Form of study
full-time

Requirements
compulsory

## Number of hours

Lecture
16

Laboratory classes
16

Other
0

Tutorials
0

Projects/seminars
0

## Number of credit points

3,00

## Coordinators

dr hab. inż. Maciej Komosiński prof. PP
maciej.komosinski@put.poznan.pl

## Lecturers

## Prerequisites

Students starting this course should have basic knowledge of computational complexity, machine learning algorithms and artificial neural networks. They should have the ability to model and solve simple optimization problems, programming skills and the ability to obtain information from provided sources. They should also understand the need to expand their competences. Moreover, in terms of social competences, students should exhibit honesty, responsibility, perseverance, cognitive curiosity, creativity, personal culture, and respect for other people.

## Course objective

1. To provide knowledge on biologically inspired optimization algorithms. 2. To present the role of biological mechanisms used in computation, their advantages and disadvantages. 3. To demonstrate common features and a unified approach to all optimization algorithms. 4. To present the field of artificial life and explain simulation models of biological systems. 5. To develop skills in efficient implementation and evaluation of the optimization algorithms – both in terms of running time and quality. 6. To learn to draw conclusions from self-conducted research, create reports on computational experiments and properly visualize the results.

## Course-related learning outcomes

Knowledge:
has a structured and theoretically founded general knowledge related to key issues in biologically inspired algorithms [k2st_w2]
has advanced detailed knowledge regarding simulations and modeling of biological phenomena [k2st_w3]
has knowledge about development trends and the most important cutting edge achievements in computer science and other selected and related scientific disciplines [k2st_w4]
has advanced and detailed knowledge of the processes occurring in the life cycle of hardware or software information systems [k2st_w5]
knows advanced methods, techniques and tools used to solve complex engineering tasks and conduct research in biologically-inspired algoritms [k2st_w6]

Skills:
is able to obtain information from literature, databases and other sources (both in polish and english), integrate them, interpret and critically evaluate them, draw conclusions and formulate and fully justify opinions [k2st_u1]
is able to plan and carry out experiments, including computer measurements and simulations, interpret the obtained results and draw conclusions and formulate and verify hypotheses related to complex engineering problems and simple research problems [k2st_u3]
can use analytical, simulation and experimental methods to formulate and solve engineering problems and simple research problems [k2st_u4]
can - when formulating and solving engineering tasks - integrate knowledge from different areas of computer science (and if necessary also knowledge from other scientific disciplines) and apply a systemic approach, also taking into account non-technical aspects [k2st_u5]
is able to assess the suitability and the possibility of using new achievements (methods and tools) and new it products [k2st_u6]
can carry out a critical analysis of existing technical solutions and propose their improvements (streamlines) [k2st_u8]
is able - using among others conceptually new methods - to solve complex it tasks, including atypical tasks and tasks containing a research component [k2st_u10]

Social competences:
understands that in the field of it the knowledge and skills quickly become obsolete [k2st_k1]
understands the importance of using the latest knowledge in the field of computer science in solving research and practical problems [k2st_k2]
is aware of the need to develop professional achievements and comply with the rules of professional ethics [k2st_k4]

## Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Formative assessment:
a) lectures:
- based on answers to questions about the material discussed in previous lectures,
b) laboratories:
- based on an assessment of the current progress in the implementation of tasks.

Summative assessment:
a) lectures: verification of the learning outcomes is carried out by:
- assessment of the knowledge and skills demonstrated in a test consisting of several questions or short tasks. Exceeding 50% of the points allows to obtain a satisfactory grade.
- discussion of the test results,
b) laboratories: verification of the learning outcomes is carried out by:
- assessment of skills related to the implementation of laboratory exercises,
- continuous assessment during each class (oral answers) - rewarding the increase in the ability to use the learned principles and methods,
- evaluation of reports prepared partly during the classes and partly after their completion, with the possibility of using the Moodle platform,
- presenting the results of individual experiments.
Obtaining additional points for activity during classes, especially for:

- carrying out extended, non-obligatory experiments as part of laboratory tasks and describing them in the report,
- remarks to improve teaching materials.

## Programme content

The classes cover evolutionary algorithms, including selection, crossover, mutation, and fitness scaling techniques. Variants like genetic algorithms, evolutionary strategies, differential evolution, and genetic programming are explored. Co-evolutionary architectures, both cooperative and competitive, are studied along with their problems and ways to mitigate them. Evolutionary design principles, genotype-phenotype mapping and modularity are introduced. Optimization challenges, fitness landscape analysis, epistasis detection, and search space transformations are also discussed.

## Course topics

Lectures:
Architecture and evaluation of evolutionary algorithms. Techniques of selection, crossover, mutation, scaling the fitness of solutions. Schema theorem and epistasis. Messy genetic algorithm. Hierarchical genetic algorithm. Mechanisms inspired by nature in evolutionary algorithms. Evolutionary strategies. Differential evolution. Evolutionary programming: floating point representation, embryogenesis, genetic operators and global convexity. Genetic programming and symbolic regression. Neutrality and genetic drift. Quality-diversity techniques. Co-evolutionary architectures – cooperative and competitive. Problems and pathologies in co-evolution and ways to reduce them. Spontaneous and directed evolution. Closed-ended versus open-ended evolution. Evolutionary design and evolutionary robotics; genotype-phenotype mapping, morphogenesis and modularity.

Laboratories:
- Comparison of optimization difficulty and discussion: classical combinatorial problems vs. evolutionary design,
- Mutation intensity and its impact on the optimization process,
- Modifying the fitness landscape, making it smoother, creating a gradient,
- Modifying the way the topology of solutions is searched. Evolution of designs and their control,
- Discovering the fitness landscape: measures of ruggedness and convexity,
- Detecting, estimating and visualizing epistasis,
- Transformation between search spaces: from the GP space to the evolutionary design space,
- Case study: optimize a shape or behavior.

## Teaching methods

Lecture: multimedia presentation illustrated with examples.
Laboratory exercises: presentation illustrated with examples; carrying out the tasks given by the teacher (practical exercises).

## Bibliography

Basic
1. D.E. Goldberg, Algorytmy genetyczne i ich zastosowania, WNT, Warszawa, 2009.
2. Z. Michalewicz, Algorytmy genetyczne + struktury danych = programy ewolucyjne, WNT, Warszawa, 2003.

Additional
1. Jason Brownlee, Clever Algorithms: Nature-Inspired Programming Recipes, 2012. https://github.com/clever-algorithms/CleverAlgorithms
2. El-Ghazali Talbi, Metaheuristics: From Design to Implementation, Wiley, 2009.

## Breakdown of average student's workload

| | Hours | ECTS |
|---|---|---|
| Total workload | 75 | 3,00 |
| Classes requiring direct contact with the teacher | 32 | 1,50 |
| Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation) | 43 | 1,50 |