A0M33EOA: Competencies

Petr Pošík, Jiří Kubalík

January 16, 2017

Abstract

List of competencies students shall gain after completing course *Evolutionary Optimization Algorithms* taught at Czech Technical University in Prague, Dept. of Cybernetics, primarilly for (but not limited to) students of Open Informatics study programme.

Prerequisities. Optimization: problems and methods.

Before entering this course, a student shall be able to

- define an optimization task in mathematical terms; explain the notions of search space, objective function, constraints, etc.; and provide examples of optimization tasks;
- describe various subclasses of optimization tasks and their characteristics;
- define exact methods, heuristics, and their differences;
- explain differences between constructive and generative algorithms and give examples of both.

1 Black-box optimization. Local search, and evolutionary algorithms.

After this lecture, a student shall be able to

- describe and explain what makes real-world search and optimization problems hard;
- describe black-box optimization and the limitations it imposes on optimization algorithms;
- define a neighborhood and explain its importance to local search methods;
- describe a hill-climbing algorithm in the form of pseudocode; and implement it in a chosen programming language;
- explain the difference between best-improving and first-improving strategy; and describe differences in the behaviour of the resulting algorithm;
- enumerate and explain the methods for increasing the chances to find the global optimum;
- explain the main difference between single-state and population-based methods; and name the benefits of using a population;
- describe a simple EA and its main components; and implement it in a chosen programming language.

2 Standard EAs. Schema theorem. Representations.

After this lecture, a student shall be able to

- know and actively use the terminology inspired by biology;
- explain the importance, role and types of representation, selection, and genetic operators;
- distinguish between premature convergence and stagnation of EA, and suggest methods to fight them;
- explain the trade-off between exploration and expoitation;
- implement 1- and 2- point crossover, uniform crossover, bit-flip mutation;
- describe options for encoding the solutions of traveling salesperson problem, and the relevant genetic operators;
- give examples of real-world problems EAs have been applied to;

- describe what a schema theory is and what it is used for;
- explain the schema theorem, and the influence of its individual parts related to selection, crossover and mutation;
- state the so-called Building Block Hypothesis and comment on its relation to the chosen representation.

3 EAs for real-parameter optimisation. Evolution strategies, CMA-ES.

After this lecture, a student shall be able to

- perform the mapping of chromosomes from binary to real space when using binary encoding for real-parameter optimization;
- describe and exemplify the effects of such a genotype-phenotype mapping on the neighborhood structures induced by mutation and crossover;
- give examples and describe some mutation and crossover operators designed for spaces of real number vectors;
- explain the main features of ES and differences to GAs;
- explain the notation $(\mu/\rho \overset{+}{,} \lambda)$-ES;
- describe the differences between mutation with isotropic, axis-parallel, and general Gaussian distribution, including the relation to the form of the covariance matrix, and the number of parameters that must be set/adapted for each of them;
- explain and use two simple methods of mutation step size adaptation (1/5 rule and self-adaptation);
- write a high-level pseudocode of CMA-ES and describe CMA-ES in the $(\mu/\rho \overset{+}{,} \lambda)$ notation;
- implement DE algorithm;
- explain the basic forms of DE mutation and crossover.

4 Genetic programming. Fundamentals and applications.

After this lecture, a student shall be able to

- explain the main differences between GA and GP, and name typical application areas for GP;
- describe the representation that GP uses, including the associated crossover and mutation operators;
- explain how GP deals with real-valued constants in evolved solutions;
- explain two different ways how GP deals with the possibility that a crossover or mutation operator results in an invalid offspring;
- describe solution initialization methods used in GP (full, grow, ramped half-n-half, PTC);
- explain greedy over-selection operator and why it was invented;
- motivate and describe the semantic crossover operators;
- explain "automatically defined functions" and motivate them;

5 Epistasis. Estimation-of-Distribution Algorithms.

After this lecture, a student shall be able to

- explain what an epistasis is and show an example of functions with and without epistatic relations;
- demonstrate how epistatic relationships can destroy the efficiency of the search performed by an optimization algorithm, and explain it using schemata;
- describe an Estimation-of-Distribution algorithm and explain its differences from an ordinary EA;
- describe in detail and implement a simple UMDA algorithm for binary representations;
- understand, fit to data, and use simple Bayesian networks;
- explain the commonalities and differences among EDAs not able to work with any interactions (PBIL, cGA, UMDA);

- explain the commonalities and differences among EDAs able to work with only pairwise interactions (MIMIC, COMIT, BMDA);
- explain the commonalities and differences among EDAs able to work with multivariate interactions (ECGA, BOA);
- explain the model learning procedures used in ECGA and BOA;
- understand what effect the use of a more complex model has on the efficiency of the algorithm when used on problems with increasingly hard interactions.

6 No Free Lunch. Empirical comparisons of stochastic optimization algorithms.

After this lecture, a student shall be able to

- explain No Free Lunch Theorem, and its consequences;
- explain the concepts of success probability, runtime distribution, solution quality, and their relationship;
- define r-complete, asymptotically r-complete, and r-incomplete algorithms;
- describe 3 usual scenarios of applying an algorithm to an optimizaton problem, and explain their differences;
- explain differences between Monte Carlo and Las Vegas algorithms;
- name the advantages and disadvantages of measuring time in seconds vs measuring time in the number of performed operations;
- explain what errorneous conclusions can be drawn from the results of an experiment when comparing algorithms using a single time limit, and/or a single required target level;
- know a few statistical test that can be used to compare 2 algorithms;
- exemplify what kind of characteristics we can get when taking cross-sections of the runtime distribution function;
- explain how the runtime distributions can be aggregated over different target levels, different problem instances and different problems;
- derive valid conclusions when presented with runtime distributions of two or more algorithms.

7 Multi-Objective Evolutionary Algorithms.

After this lecture, a student shall be able to

- define a multi-objective optimization problem and describe the relationship between decision and objective spaces;
- define the dominance principle and the Pareto-optimal solutions;
- identify non-dominated solutions in a set of solutions;
- list and describe two goals of multi-objective optimization;
- describe some non-evolutionary approaches to multi-objective optimizatin and explain their deficiencies;
- implement evolutionary multi-objective algorithms and explain their differences from ordinary EA;
- explain algorithms NSGA, NSGA-II, SPEA2 and their differences;
- implement constraint handling in NSGA-II;
- define performance measures used in multi-objective optimizations (S metric and C metric);

8 Constraint-Handling in Evolutionary Algorithms.

After this lecture, a student shall be able to

- define the general nonlinear programming problem with constraints;
- define the general form of fitness function with penalty function;
- describe the main categories of constraint handling approaches with penalty function – death penalty, static penalty, dynamic penalty, adaptive penalty – and explain their shortcomings;

- describe the GA with non-linear penalty function and its main characteristics;
- list and describe the principal components of the Adaptive Segregational Constraint Handling EA (ASCHEA);
- explain the main idea behind the Stochastic Ranking adaptive penalty approach;
- implement random keys representation used for solving permutation problems;
- explain the main idea behind multi-objective approaches to constraint handling.

9 Grammatical Evolution (GE). Cartesian Genetic Programming (CGP).

After this lecture, a student shall be able to

- implement a variable-length linear representation and a genotype-phenotype mapping used in GE;
- describe a representation of a program in CGP in the form of a directed graph;
- explain the neutral mutations in GE and CGP and their effect on the search process;
- describe the ripple crossover in GE;
- write a high-level pseudocode of GE and CGP;
- implement a concept of automatically defined functions into GE (Grammatical Evolution or meta-Grammar GE, GE grammar with the ability to define one more ADFs);
- explain the explicit automatic code reuse in CGP;

10 Gene Expression Programming (GEP). Bloat in GP.

After this lecture, a student shall be able to

- describe the fixed length linear representation and the genotype-phenotype mapping used in GEP;
- explain how tree structures of varying size and shape are evolved using the fixed length linear representation;
- explain the structure of a gene in GEP, i.e. the head and tail parts and its sizing;
- describe multigenic chromosomes used to evolve modular structures by GEP;
- describe how constants are handled in GEP;
- explain genetic operators used in GEP;
- define the phenomenon called a code bloat and its effects on the search process of GP;
- describe so-called introns and their role in code bloat;
- list and explain some reasons for why the code bloat happens;
- describe two variants of the Lexicographic Parsimony Pressure method and their characteristics;
- describe the Dynamic Size Limits method and its characteristics;
- describe the Operator Equalisation method and its characteristics;

11 Parameter Tuning.

After this lecture, a student shall be able to

- identify metaparameters (tunable parameters) in various optimization tasks and distinguish them from decision variables;
- explain the difference between *parameter tuning* and *parameter control*, and give examples of both;
- define the task of parameter tuning;
- explain the complex nature of the parameter tuning problem and describe characteristics that make it complex;
- list several contributions of parameter tuning;
- exemplify a few manual methods usable for parameter tuning, list their advantages and disadvantages;
- describe and explain racing techniques, F-race and iterated racing, and its advanatges/disadvantages;

- describe and explain ParamILS algorithm (how the local search is done, how the *iteration* of local search is done) + its advantages/disadvantages;
- explain the principle of using surrogate models in optimization and describe possible shortcommings;
- describe Gaussian process and explains its difference to the majority of regular regression models;
- explain the role of an *acquisition function* in Gaussian process-based optimization and list a few examples of acq. functions.

12 Parallel EAs.

After this lecture, a student shall be able to

- explain the main motivation for parallelization;
- explain the difference between parallel *implementation* and parallel *model* of EA;
- describe the features of individual combinations of sequential/parallel implementation and global/parallel model;
- know which parts of individual EAs (fitness evaluation, selection, replacement, mutation, crossover, model building, etc.) can be implemented in parallel easilly, and explain why;
- explain the principle and features of the master-slave parallelization;
- implement island model and explain its features, describe the characteristics of migration operator (type, topology, degree of connectivity, trigger, selection/replacement strategy, count);
- describe spatially embedded model, heterogeneous model, injection model, and their use cases.

13 Coevolution.

After this lecture, a student shall be able to

- define “coevolution”, explain what makes it different from ordinary evolution in the context of optimization algorithms;
- explain differences between *cooperative* and *competitive* coevolution;
- explain differences between *intra-* and *inter*-population coevolution;
- define features of fitness measure (static/dynamic, deterministic/stochastic, absolute/relative, internal/external);
- describe the features of an ideal external (and internal) fitness, and describe the features of internal fitness in coevolution;
- exemplify individual types of cooperative/competitive intra-/inter-population coevolution; and
- exemplify various types of issues that can be observed in them (how to order individuals based on inconsistent matches, loss of diversity when one population evolves faster than the other, hijacking, relative overgeneralization, miscoordination).