Projects - Metaheuristic Algorithms

2015-2016

I. Application oriented topics

1. **Task scheduling in distributed systems.** The aim is to assign a set of (independent or correlated) tasks to some machines such that some quality measures (e.g. total execution time, lateness etc) are optimized. There are a lot of variants of this problem and it can be solved by using various heuristics and meta-heuristics: simulated annealing (SA), evolutionary algorithms (EA), ant colony optimization (ACO), particle swarm optimization(PSO).

   Requirement: Select a problem instance (there are sets of test problems, e.g. Braun benchmark: testData_512x16.zip); describe an adequate method (based on SA, EA, ACO or PSO), implement and test it.

   Variants: 4 (SA, EA, ACO, PSO)

   Biblio: aplicatii/task_scheduling

2. **Cloud Resource Provisioning.** It refers to provide computational resources (e.g. virtual machines) for a specific task such that some quality of service requirements (storage, cost, energy consumption etc). The problem can be addressed from the point of view of the provider (efficiently allocate virtual machines on physical resources) and from the point of view of the user (select the best services). In both cases it can lead to constrained, multi-objective and/or dynamic optimization problems for which various metaheuristics have been proposed lately. To test various strategies one can use CloudSim (https://github.com/Cloudslab/cloudsim) which is a cloud simulation framework which allows to simulate data centers, virtual machines and various policies for resource allocation and energy consumption control.

   Requirement: Select a problem and a method, implement the method and test it using CloudSim.

   Variants: 4 (EA, ACO, PSO, other metaheuristics)

   Biblio: aplicatii/CloudProvisioning

   CloudSim: https://github.com/Cloudslab/cloudsim

3. **VRP - Vehicle Routing Problem.** It is a combinatorial optimization problem which is a generalization of TSP. It appears in different real world situations and can be solved by using evolutionary algorithms, ant-based methods or other meta-heuristics. There are different variants of the problem (see for instance http://neo.lcc.uma.es/vrp/vrp-flavors/).

   Requirement: Select a problem instance, describe an adequate method, implement and test it for one of the instances available at http://neo.lcc.uma.es/vrp/

   Variante: 4 (EA, TS, PSO, VNS)

   Biblio: aplicatii/VehicleRouting

4. **Nurse rostering.** This is a practical problem asking to find a schedule of nurses in a hospital. It is in fact a constraint satisfaction problem which can be solved by using different meta-heuristics (evolutionary algorithms (EA), estimation of distribution algorithms (EDA), ant colony optimization (ACO), variable neighborhood search (VNS), tabu search (TS), hyperheuristics etc).

   Requirement: Select a problem variant, identify and describe an adequate method (based on EA, EDA, ACO, VNS or TS), implement and test it.

   Variants: 5 (EA, EDA, ACO, VNS, TS)

   Biblio: aplicatii/nurse_rostering

5. **Timetabling.** It is also a constraint satisfaction problem characterized by the existence of hard constraints (they should be satisfied) and soft constraints (they can be broken but the score of the timetabling becomes smaller). There are approaches based on simulated annealing (SA), evolutionary algorithms (EA), ant colony optimization (ACO), tabu search (TS), variable neighbourhood search (VNS), particle swarm optimization (PSO)

   Requirement: Select a problem variant, identify and describe an adequate method (based on SA, EA, ACO, TS, VNS or PSO), implement a variant and test it.

   Variants: 6 (SA, EA, ACO, TS, VNS or PSO)

   Biblio: aplicatii/timetabling

6. **Credit scoring.** Credit scoring is a classification problem aiming to discriminate the credits which involves risks for the bank from those which do not present risks. There are several approaches in solving this problem: neural networks (NN), genetic programming (GP), artificial immune systems (AIS).

   Requirement: Identify and describe an adequate method (based on NN,GP, AIS), implement a variant and test it.

   Variants: 3 (NN, GP, AIS)

   Biblio: aplicatii/credit_scoring

7. **Portfolio optimization.** Portfolio optimization means to find financial or production portfolios which maximize the profit and minimize the risk. The problem can be formulated as a constrained combinatorial optimization problem which can be solved using various metaheuristics (e.g. GA, GP, PSO, Hopfield Neural Networks - HNN)

   Requirement: Identify and describe an adequate method (based on GA, GP, PSO, HNN), implement a variant and test it.

   Variants: 4 (GA, GP, PSO, HNN)

   Biblio: aplicatii/portfolio_optimization

8. **Community detection in networks.** The aim is to identify the groups of highly interconneted nodes in a network. The problem is similar to that of graph partitioning such that the nodes in a partition are highly connected while the nodes in different partitions are sparsely connected. It can be solved by using clustering techniques but also evolutionary algorithms.

   Requirement: Study aof the community detection problem, implementation of an EA and its testing for a test problem (e.g. Zachary's club problem).

   Biblio: aplicatii/communities_detection

9. **Intrusion detection systems.** Intrusion detection systems (IDS) are mainly based on classifying the behaviour of the user of a computing system in normal and abnormal, based on data related to the access of the user to the computing resource. IDSs usually monitor all events taking place in the computing system. Currently there are various approaches based on computational intelligence (CI) techniques: neural networks (NN), evolutionary algorithms (EA) and artificial immune systems (AIS). Current malware detection products (e.g. Bitdefender) incorporate various machine learning algorithms (e.g. perceptron-like classifiers) or metaheuristics of relevant attribute selection.

   Requirement: Analyze the structure of a IDS and implement at least one component based on a CI technique: NN, EA or AIS.

   Variants: 3 (NN, EA, AIS)

   Biblio: aplicatii/intrusion_detection_systems

10. **Software testing.** The evolutionary algorithms and other nature inspire metaheuristics (e.g. ACO) can be used to generate test cases for software analysis.

    Requirement: Study of the applicability of evolutionary algorithm in software testing. Choose an evolutionary algorithm to generate test cases for a software product and implement it.

    Biblio. aplicatii/software_testing

11. **Automated Program Repairing.** Automatic generation of patches used to remove bugs from software or to make an application more efficient can be done using genetic programming. The patches are usually small pieces of code which are generated starting from some templates and are evaluated using the behavior of the program for some testcases.

    Requirement: Comparative study of different approaches in automated patches generation. A Genetic Programming based example should be implemented.

    Biblio. aplicatii/AutomatedProgramRepair

12. **Parameter tuning.** The behaviour of different software tools may depend on their control parameters and finding the appropriate parameter values can lead to difficult combinatorial optimization problems for which metaheuristics could be useful. Such cases may happened in search based software engineering, in tuning parameters of image processing tools (e.g. for registration, segmentation) or tuning the parameter of various physical models (e.g. weather models).

    Requirement: Select a problem and a metaheuristic and implement a parameter tuning example.

    Biblio. aplicatii/ParameterTuning

13. **Data analysis and prediction.** Lately, in the context of major conferences in evolutionary computing (e.g. GECCO, CEC) have been organized competitions on industrial challenges, most of them addressing prediction problems.

    Requirement: Implementing a solution to one of the problems from:

    - GECCO 2013 (http://www.spotseven.de/gecco-challenge/gecco-challenge-2013/)
    - GECCO 2014 (http://www.spotseven.de/gecco-challenge/gecco-challenge-2014/)
    - GECCO 2015 (http://www.spotseven.de/gecco-challenge/gecco-challenge-2015/)

II. Topics oriented towards techniques

1. **ABC - Artificial Bee Colony.** It is a metaheuristic inspired by the behaviour of bee colonies (how they find food or communicate). Basic info can be found at http://mf.erciyes.edu.tr/abc/ *Requirement:* A review of ABC techniques, implementation of a ABC algorithm and testing for a continuous or a combinatorial optimization problem.

   Biblio. tehnici/ABC

2. **ACO - Ant Colony Optimization, AS - Ant systems.** These are metaheuristics inspired by the behaviour of ants. The main characteristic is the existence of an indirect communication between the individuals of the same colony based on the pheromone trails. They can be used in solving routing or scheduling problems.

   Requirement. Study of ant-based algorithms, implementation of a variant and its application for a problem (routing, scheduling or assignment).

   Biblio. tehnici/ACO

3. **BBO - Biogeography Based Optimization.** This is a recent metaheuristic inspired by the properties of the geographical distribution of biological organisms. It uses a particular crossover operator inspired by the emigration/immigration processes.

   Requirement. Study of BBO, implementation of a variant and its application for a simple optimization problem.

   Biblio. tehnici/BBO

4. **PSO - Particle Swarm Optimization** This is inspired by the behavior of birds swarms or of other social entities. Each individual in the swarm decides which is its next position in the search space based both on the personal experience and the experience of the swarm. It can be used for solving continuous optimization problems.

   Requirement: Study of PSO methods, implementation of a variant and application for an optimization problem.

   Biblio. tehnici/PSO

5. **VNS - Variable Neighborhood Search** It is a search technique involving selection of perturbed configurations inside a neighborhood of the current configuration.

   Requirement: Study of VNS methods, implementation of a variant and application for an optimization problem.

   Biblio. tehnici/VNS

6. **Cooperative Coevolution.** Coevolution means the simultaneous evolution of several populations. Cooperative coevolution provides a general framework for solving complex problems by dividing them in simpler problems. This strategy can be combined with any type of evolutionary algorithms and has been successfully applied for high-dimensional optimization problems.

   Requirement. Study of cooperative coevolution, implementation of a coevolutionary algorithms and its application for a optimization problem with a large number of variables (at least 100).

   Biblio. tehnici/coevolution

7. **Compact Evolutionary Algorithms.** They are EAs designed under the assumption that there is a limited amount of resources available for their execution (e.g. they should be executed on mobile devices). In order to deal with this problem they do not use explicit populations but only descriptions of the probability distributions which models the elements of the populations.

   Requirement. Study of compact EAs, implementation of such an algorithm and its application for a simple optimization problem.

   Biblio. tehnici/compactEA

8. **EDA - Estimation of Distribution Algorithms.** They are stochastic optimization techniques that uses probabilistic models for exploring the search spaces (they are somewhat similar to compact EAs). *Requirement*: Study of EDA algorithms, implementation of a variant and testing for a simple optimization problem (e.g. Onemax).

   Biblio. tehnici/EDA

9. **Hyperheuristics.** They are heuristics which "chooses heuristics" in order to design a method for solving a problem. The selection of heuristics can be random, greedy, based on some learning processes or even based on genetic algorithms. They have succesfully applied for combinatorial optimization problems.

   Requirement: Study of hyperheuristics, implementation of a variant and testing it for a simple optimization problem.

   Biblio. tehnici/hyperheuristics

10. **Cuckoo Search.** This technique is inspired by the cuckoo behaviour (the use the nests of other species to place their eggs). This idea is modelled by particular mutations involving Levy distributions.

    Requirement. Study of the state-of-the-art of cuckoo search, software implementation and testing for some optimization problems. *Biblio.* tehnici/CuckooSearch

11. **Firefly algorithms.** This technique is inspired by the fireflyes "lightening" behavior. The biological inspiration is modelled through attraction/repelling mechanisms which control the search process.

    Requirement. Study of the state-of-the-art of fireflies algorithm, software implementation and testing for some optimization problems.

    Biblio. tehnici/FireflyAlgorithm

12. **Bat algorithms.** This metaheuristic is isnpired by the echo-location mechanism used by bats in the orientation process. The idea is modelled by combining a local search with a random search.

    Requirement. Study of the state-of-the-art of bat algorithm, software implementation and testing for some optimization problems.

    Biblio. tehnici/BatAlgorithm

13. **Deep Learning** Since introduction in 2006 deep learning became a powerful tool in solving handwriting recognition, image recognition, speech recognition and lately in natural language processing (see the Google tool word2vec). These models are based on complex architectures

which contain recurrent subnetworks (e.g. Boltzmann machines) and on unsupervised pre-training.

Requirement. Study of Deep Learning models and of the tools Theano (http://deeplearning.net/software/theano/), PyLearn (http://deeplearning.net/software/pylearn2/) and word2vec (https://code.google.com/p/word2vec/).

Biblio. tehnici/DeepLearning

14. **Reservoir computing.**
Reservoir computing (http://reservoir-computing.org/, http://reslab.elis.ugent.be/) is an approach to train recurrent neural networks used in processing time series (classification or prediction). These models contain some random hidden connections and only the weights corresponding to the connections leading to output units are trained.

    Requirement. Study of at least one reservoir computing model (Echo State Network, Liquid State Machine etc) and a small example implementation.

    Biblio. tehnici/ReservoirComputing

15. **GPU implementation of EAs.** Implementation of EAs on GPU became popular in the last years. The main issue is related to the choice of operation to be done on GPU such that the transfer between CPU and GPU is minimized.

    Requirement: GPU implementation of a simple GPU.

    Biblio. aplicatii/EA+GPU

III. Comparative studies of neural evolutionary tools.

Examples of tools which can be compared:

Neural Computation:

1. **Neuroph** - a lightweight Java framework for developing neural networks. (Java) http://goodoldai.org/neuroph
2. **FANN** - free open source neural network library, which implements multilayer artificial neural networks in C with support for both fully connected and sparsely connected networks.(C++) http://leenissen.dk/fann/wp/
3. **PyNN** - a simulator-independent language for building neuronal network models. (Python) http://neuralensemble.org/trac/PyNN
4. **PyBrain** - a modular Machine Learning Library for Python. (Python) http://pybrain.org/pages/home
5. **HyperNEAT.** (C++) http://eplex.cs.ucf.edu/hyperNEATpage/HyperNEAT.html
6. **ENCOG** - an advanced machine learning framework. (C++, Java) http://www.heatonresearch.com/encog
7. **SimBrain** - a free tool for building, running, and analyzing neural-networks. (Java) http://www.simbrain.net/
8. **TensorFlow** - an open source software library for numerical computation using data flow graphs developed by the Google Brain Group. (Python, C++) https://www.tensorflow.org/

Evolutionary Computation:

1. **ECJ.** http://cs.gmu.edu/ eclab/projects/ecj/
2. **JGap.** http://jgap.sourceforge.net/
3. **JavaEVA.** http://www.ra.cs.uni-tuebingen.de/software/EvA2/
4. **JCLEC.** http://jclec.sourceforge.net/
5. **JaGA.** http://www.jaga.org/
6. **JMetal.**http://jmetal.sourceforge.net/
7. **GAA.** http://www.aridolan.com/ga/gaa/gaa.html
8. **EO.** http://eodev.sourceforge.net/
9. **ParadisEO.** http://paradiseo.gforge.inria.fr/index.php?n=Main.HomePage
10. **PISA.** http://www.tik.ee.ethz.ch/sop/pisa/
11. **OpenBeagle.** https://github.com/chgagne/beagle
12. **Robust Genetic Programming.** http://robgp.sourceforge.net/about.php
13. **DEAP - Distributed Evolutionary Algorithms in Python** http://code.google.com/p/deap/
14. **pySTEP - Python Strongly Typed gEnetic Programming** http://pystep.sourceforge.net/
15. **GPE - Genetic Programming Engine** http://gpe.sourceforge.net/
16. **PerlGP** - Perl Genetic Programming System http://perlgp.org/
17. **MetsLib** - metaheuristic modeling framework and optimization toolkit in modern C++ https://projects.coin-or.org/metslib
18. **jmp75** - a simplified metaheuristics framework for .NET. https://github.com/jmp75/metaheuristics
19. **HEO** - cross-platform library that provides various parallel metaheuristic algorithms for hard optimization problems. https://code.google.com/p/heo/
20. **FOM** - a Framework for Optimization using Metaheuristics. http://www.isa.us.es/fom/modules/portalWFInterface/init.php
21. **OPT4J** - open source Java-based framework for evolutionary computation. http://opt4j.sourceforge.net/
22. **HeuristicLab** - a framework for heuristic and evolutionary algorithms. http://dev.heuristiclab.com/

Requirement. At least two tools should be selected, described and tested for a simple problem. The project will contain a comparative study of the selected tools.

Structure of the project. Each project will consist of:

1. A report (6-12 pages) structured as follows:
   - abstract (5 lines): will describe the aim of the report and the main result
   - introduction: will contain a state of the art concerning the studied method/application; it is based on the study of the papers specified as references
   - detailed presentation of the models/algorithms/problems related to the subject
   - detailed presentation of the application + results
   - conclusions and open problems
   - references
2. A functional application illustrating the behaviour of the method(s) described in the report or using the software tools selected for comparison.

Remarks.

- The papers suggested for references can be dowloaded from
  http://www.info.uvt.ro/~dzaharie/am2015/proiecte

  These papers should be used to start the study - other references could be used as well. The report should be based on at least two references.
- The report should not contain paragraphs directly copied from other papers. The existence of such paragraphs is considered to be plagiarism and such a report cannot be accepted. In order to avoid plagiarism you should synthesize the main ideas using you own words.