

Teme pentru proiecte - Algoritmi metaeuristici

2017-2018

I. Teme orientate înspre aplicații

Specificul acestor teme este faptul că se pornește de la o problemă concretă și trebuie identificată și implementată o metodă de rezolvare. Metodele vizate se bazează pe diferite tipuri de metaeuristici (inclusiv algoritmi evolutivi și alte metaeuristici inspirate de natură) precum și modele bazate pe învățare automată, cum sunt rețelele neuronale.

1. **Solvere bazate pe algoritmi euristici pentru rezolvarea de probleme liniare (Integer Linear Programming) (IACD, IASTE)** Problemele care se rezolvă folosind Integer Linear Programming sunt probleme de optimizare combinatorială. Acestea presupun realizarea de optimizări pentru funcții liniare care sunt supuse unor constrângeri. Linear Integer Programming permite rezolvarea problemelor utilizând algoritmi de tip simplex.

Cerința: Selecția unei metode, implementarea metodei și testarea acesteia folosind MIPLIB (Mixed Integer Programming LIBrary - <http://miplib.zib.de/>).

Variante: 2 (EA, TS)

Biblio: aplicatii/LiniarIntegerProgramming-Evolutionary

2. **Alocarea resurselor în cloud - Cloud Resource Provisioning (IACD, IS).** Se consideră problema asignării de resurse cloud pentru o aplicație specifică astfel încât să fie satisfăcute cerințe privind caracteristicile resurselor de calcul și restricții privind costul. Problema poate fi abordată din cel puțin două perspective: cea a utilizatorului (conducând la problema selecției mașinilor virtuale) și cea a provider-ului (conducând la problema mapării mașinilor virtuale pe mașini fizice). În ambele situații se poate ajunge la rezolvarea unor probleme de optimizare care implică restricții și/sau mai multe criterii de optim, pentru care au fost propuse în ultimii ani diferite metaeuristici. Pentru testarea diferitelor strategii se poate folosi CloudSim (<https://github.com/Cloudslab/cloudsim>) care e un framework ce permite modelarea și simularea unei infrastructuri de tip cloud oferind suport pentru simularea de mașini virtuale, centre de date, diferite politici de alocare a resurselor precum și controlul consumul de energie.

Cerința: Selecția unei probleme și a unei metode, implementarea metodei și testarea folosind CloudSim.

Variante: 4 (EA, ACO, PSO, alte metaeuristici)

Biblio: aplicatii/CloudProvisioning, CloudSim: <https://github.com/Cloudslab/cloudsim>

3. **Optimizarea traseelor - Vehicle Routing Problem (IACD, IASTE).** VRP (Vehicle Routing Problem) este o generalizare a problemei comis voiajorului și are ca scop optimizarea traseelor mai multor mijloace de transport care asigură aprovizionarea mai multor clienți cu

produse preluate de la unul sau mai multe depozite. Există numeroase variante ale problemei care diferă între ele prin restricțiile impuse (ex: capacitatea limitată a vehiculelor, intervalele de timp în care trebuie realizată aprovizionarea, modificarea cerințelor în mod dinamic etc) sau prin criteriul/criteriile de optim (cost, distanță parcursă, număr clienți serviți etc). Exemple de astfel de variante pot fi găsite la <http://neo.lcc.uma.es/vrp/vrp-flavors/>. VRP poate fi descompusă în probleme de selecție, alocare, împachetare, ordonare și poate fi rezolvată utilizând diferite tipuri de euristici și metaeuristici.

Cerința: Selecția unei probleme și a unei metode, implementarea metodei și testarea pentru una din instanțele de la <http://neo.lcc.uma.es/vrp/>

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

Biblio: aplicatii/VehicleRouting

4. **Planificarea traseelor - Path finding Problem (IACD, IASTE).** Problema de path finding apare în contextul roboțiilor inteligenți care trebuie să identifice și evite obstacole pentru a găsi cea mai scurtă cale de la o sursă la destinație.

Cerința: Selecția unei probleme și a unei metode, implementarea metodei și testarea ei

Variante: 4 (EA, PSO, ABC, ACO)

Biblio: aplicatii/pathFinding

5. **Planificarea turelor într-un spital - Nurse rostering (IACD, IASTE).** Unul dintre exemplele cel mai mult studiate este cel al planificării turelor asistentelor medicale dintr-un spital. Este o problemă de satisfacere a restricțiilor care poate fi rezolvată cu meta-euristici de tip simulated annealing (SA), algoritmi genetici, estimation of distribution (EDA), ant colony optimization (ACO), artificial bee colony (ABC), variable neighborhood search (VNS), tabu search (TS) etc.

Cerința: Selecția unei metode adecvate (EA, SA, ACO, sau altă metodă), prezentarea metodei, implementarea și testarea pentru o instanță a problemei.

Biblio: aplicatii/nurse-rostering

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

6. **Generarea orarelor - Timetabling (IACD, IASTE).** Planificarea unei activități (de exemplu construirea unui orar) este o problemă cu numeroase restricții și unul sau mai multe criterii de optimizat. Unele restricții sunt puternice (trebuie respectate întotdeauna) iar altele sunt mai slabe (pot fi încălcate însă induc o creștere a costului - vezi tehnica penalizării). Pentru rezolvarea acestor probleme s-au dezvoltat diverse metode euristice (tabu search - TS, simulated annealing - SA, algoritmi evolutivi - EA, variable neighbourhood search - VNS, particle swarm optimization - PSO etc).

Cerința: Selecția unei metode adecvate (TS, SA, GA sau altele), prezentarea metodei, implementarea și testarea pentru o instanță a problemei (generarea unui orar).

Variante: 7 (SA, EA, ACO, ABC, TS, VNS, PSO)

Biblio: aplicatii/timetabling

7. **Optimizarea semaforizării - Traffic Lights Optimization (IS, IACD, IASTE).** Se referă la stabilirea duratei pentru fiecare dintre stările unui semafor (roșu, verde) astfel încât să fie optimizate anumite criterii (de exemplu să fie maximizat numărul vehiculelor care

ajung la destinație și/sau minimizarea timpului de deplasare per vehicul între două puncte). În contextul utilizării de metaeuristici bazate pe populații, o soluție este reprezentată ca un vector de valori care reprezintă durata în secunde corespunzătoare celor două stări (roșu și verde) corespunzătoare tuturor semafoarelor din zona analizată. Evaluarea calității unei soluții se bazează pe simularea traficului

Cerința: Studiul unui model de optimizare bazat pe metaeuristici (vezi lucrările de la bibliografie), implementarea unei variante și testarea pe un scenariu simplu (de exemplu 1-2 intersecții din Timișoara) folosind un simulator de trafic (de exemplu SUMO - http://sumo.dlr.de/wiki/Main_Page).

Biblio: aplicatii/TrafficLightsOptimization + <http://neo.lcc.uma.es/problems/traffic-lights/index.html> + http://sumo.dlr.de/wiki/Main_Page

8. **Generarea de expresii regulate - Regular Expressions Inference (IS,IACD).** Expresiile regulate sunt utile pentru extragerea de informații din text. Sinteza automată a expresiilor regulate pornind de la exemple de text poate fi realizată folosind programare genetică.

Cerința: Analiza implementării disponibile la <https://github.com/MaLeLabTs/RegexGenerator> și <http://regex.inginf.units.it>, adaptarea și testarea pentru generarea unor expresii regulate care permit extragerea unor fragmente de text (de exemplu valorile numerice dintr-un text).

Biblio: aplicatii/RegularExpressionsInference

9. **Selecția atributelor din date - Feature selection (AICD, IASTE).** Scopul urmărit este extragerea celor mai reprezentative atribute din perspectiva procesului de clasificare/grupare a datelor. Există două abordări principale: de tip filtru (independent de procesul de clasificare) și de tip wrapper (încorporat în procesul de clasificare). Problema de selecție poate fi formulată ca o problemă de optimizare uni sau multi-criterială și poate fi rezolvată folosind algoritmi evolutivi (EA), ant colony optimization (ACO), tabu search (TS) etc.

Cerința: Implementarea unui algoritm pentru selecția atributelor (de tip filtru sau wrapper combinat fie cu algoritmi de clasificare fie cu algoritmi de grupare) și testarea pentru date de la UCI Machine Learning repository (<http://www.ics.uci.edu/~mllearn/MLRepository.html>).

Variante: 3 (EA, ACO, TS)

Biblio: aplicatii/feature_selection

10. **Detectarea comunităților în rețele sociale - Community detection (IACD, IS, IASTE).** Are ca scop identificarea grupurilor de elemente puternic interconectate (de exemplu utilizatori ai unei rețele sociale). Este echivalentă cu problema partiționării grafurilor în submulțimi de noduri astfel încât nodurile din aceeași submulțime au grad mare de interconectare pe când nodurile din submulțimi diferite au grad scăzut de interconectivitate. Problema se poate rezolva aplicând tehnici de grupare (partiționale, ierarhice sau spectrale) dar și utilizând algoritmi evolutivi.

Cerința: Studiul problemei detectării comunităților, implementarea unui algoritm evolutiv (dintre cei descriși în lucrările de la bibliografie) și testarea pentru o problemă de test (de exemplu problema "Zacharys karate club network").

Biblio: aplicatii/communities_detection

11. **Testarea aplicațiilor software - Evolutionary software testing (IS, IACD)** Algoritmii evolutivi sau metaeuristicele inspirate de natura (de exemplu ACO) au fost recent utilizați în generarea automată a cazurilor de test pentru testarea structurală a produselor software.
Cerința: Studiu bibliografic al utilizării algoritmilor evolutivi în testarea produselor software și implementarea unui algoritm pentru generarea datelor de test pentru o aplicație la alegere.
Biblio. aplicatii/software_testing
12. **Evolutionary Art. (IASTE, IACD)** Algoritmii metaeuristici pot fi utilizați pentru a sintetiza imagini artistice pornind de la șabloane și aplicând algoritmi de încrucișare și mutație. În general evaluarea calității rezultatelor se bazează pe interacțiunea cu utilizatorul care poate acorda scoruri configurațiilor intermediare.
Cerința: Implementarea unui algoritm metaeuristic pentru sinteza unor imagini artistice sau a unor compoziții muzicale (la alegere).
Biblio: aplicatii/EvolutionaryArt
13. **Segmentarea imaginilor - Multilevel Thresholding** Segmentarea imaginilor presupune identificarea de regiuni în imagini astfel încât pixelii din aceeași regiune să aibă caracteristici similare iar cei din regiuni diferite să aibă caracteristici diferite. Una dintre cele mai simple tehnici de segmentare (pentru imagini pe nivele de gri) este cea bazată pe mai multe valori prag (multilevel thresholding) care se bazează pe ideea de a determina valori ale pragurilor care maximizează varianța sau entropia calculată la nivelul regiunilor.
Cerința: Implementarea unui algoritm metaeuristic (pornind de la una dintre variantele descrise în lucrările de la bibliografie) pentru estimarea valorilor unor praguri + testarea pe imagini de test (<http://sipi.usc.edu/database/>, http://www.imageprocessingplace.com/root_files_V3/image_databases.htm)
Biblio: aplicatii/ImageProcessing
14. **Analiza datelor și predicție.** În ultimii ani, în cadrul conferințelor majore din domeniul calculului evolutiv (GECCO, CEC) au fost organizate competiții de soluții la probleme provenite din industrie.
Cerința: Implementarea unei soluții la una dintre probleme enunțate la:
 - 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. Teme orientate înspre tehnici

Scopul acestor teme este de a analiza particularitățile și aplicabilitatea unor tehnici metaeuristice. Proiectul presupune un studiu bibliografic al tehnicii și implementarea unui exemplu.

1. **ABC - Artificial Bee Colony (IACD, IASTE)**. Este o tehnică inspirată de comportamentul coloniilor de albine, modul în care se organizează și de specificul comunicării dintre ele. Informații actuale despre ABC pot fi găsite la <http://mf.erciyes.edu.tr/abc/> *Cerința:* Studiu bibliografic a algoritmilor de tip ABC și implementarea unui algoritm din această categorie pentru rezolvarea unei probleme de optimizare continuă (funcție test) sau a uneia de optimizare discretă.

Biblio. tehnici/ABC

2. **AS - Ant systems (IACD, IASTE)**. Comportarea insectelor ce trăiesc în colectivități (furnicile și albinele) reprezintă sursă de inspirație pentru dezvoltarea unor metode de optimizare (pentru rezolvarea unor probleme dificile de optimizare combinatorială - problema comis voiajorului, planificarea activităților etc.) și control distribuit (de exemplu rutarea adaptivă în sistemele de comunicații). Abilitatea acestor comunități de a rezolva probleme se bazează pe comunicarea pe cale chimică (prin intermediul feromonilor): pe traseul parcurs pentru căutarea hranei furnicile lasă o "urme" care permite identificarea ulterioară a traseului care a dus la găsirea hranei. Sistemele artificiale implementează acțiunea feromonilor prin "recompensarea" configurațiilor bune și evaporarea feromonilor prin "uitarea" unor configurații.

Cerința. Studiu bibliografic al sistemelor de tipul coloniilor de furnici și implementarea unui algoritm din această clasă pentru rezolvarea unei probleme de optimizare (varianțe: (a) problema comis voiajorului, (b) probleme de orientare, (c) rutare, (d) probleme de asignare).

Biblio. tehnici/ACO

3. **BBO - Biogeography Based Optimization (IACD, IASTE)**. Este o tehnică recentă inspirată de distribuția geografică a organismelor biologice. Utilizează operatori speciali de încrucișare inspirați de procesele de emigrare și imigrare.

Cerința. Studiul BBO, implementarea unei variante și aplicarea pentru o problemă simplă de optimizare.

Biblio. tehnici/BBO

4. **PSO - Particle swarm optimization (IACD, IASTE)**. Tehnica de optimizare de tip "particle swarm optimization - PSO" are ca sursa de inspirație comportarea socială (legată în special de transmiterea și utilizarea în comun a informației) a unor ființe vii cum sunt stolurile de păsări, roiurile de albine sau bancurile de pești. În variantele artificiale procesul de căutare este asigurat de un ansamblu de "particule" a căror mișcare este caracterizată de o "viteză" care se modifică în timp în funcție de caracteristicile întregului sistem. Permite găsirea rapidă a optimului însă are dificultăți în evitarea minimelor locale. Recent au fost extinse și pentru rezolvarea problemelor de optimizare multicriterială și pentru optimizarea dinamică.

Cerința: Studiu bibliografic a algoritmilor de tip PSO și implementarea unui algoritm de tip PSO pentru rezolvarea unei probleme de optimizare continuă (funcție test) sau a uneia de optimizare discretă.

Biblio. tehnici/PSO

5. **AIS - Artificial Immune Systems (IACD, IASTE)** Sunt inspirate de sistemul imunitar natural care la apariția unui agent patogen reacționează prin proliferarea (clonarea) unor celule imunitare care "recunosc" patogenul. O parte dintre acestea devin efectoare, producând anticorpi altele au rol de memorare a "amprentei" patogenului (astfel încât la atacuri ulterioare ale aceluiași agent sistemul să reacționeze rapid). Sistemele imunitare artificiale sunt utilizate pentru rezolvarea unor probleme de clasificare (pe baza proprietății de afinitate a unei celule imunitare pentru un anumit agent), pentru optimizare și pentru detectarea atacurilor asupra calculatoarelor (de tipul virusilor).

Cerința: Studiu bibliografic al sistemelor imunitare artificiale și implementarea unui algoritm din această clasă pentru rezolvarea unei probleme de optimizare (ex: AINet sau problema comis-voiajorului) sau a altor probleme (detectie fraude, detectie intrusi in sisteme de calcul etc.).

Biblio. tehnici/AIS

6. **EDA - Estimation of Distribution Algorithms (IACD, IASTE).** EDA reprezinta o clasa de metode de optimizare bazate pe simularea distribuțiilor de probabilitate corespunzătoare configurațiilor de succes din populația curentă. S-au aplicat cu succes

Cerința: Studiu bibliografic al algoritmilor de tip EDA și implementarea unui algoritm din această clasă pentru rezolvarea unei probleme de optimizare.

Biblio: tehnici/EDA

7. **TS - Tabu search (IACD, IASTE).** Este o tehnică euristică de căutare bazată pe o listă de configurații prohibite. A fost propusă de Glover în 1986 și a fost aplicată cu succes în rezolvarea problemelor de optimizare cu restricții (de exemplu în probleme de planificare a activităților). Tabu search consta în trei faze: căutare preliminară, intensificare și diversificare. Prima etapă consta în selecția din vecinătatea unei configurații curente a celei mai bune configurații din perspectiva funcției obiectiv. Noua configurație este acceptată chiar dacă este mai rea decât cea curentă. Pentru a evita trecerea succesivă între două configurații (intrarea într-un ciclu) se gestionează o listă de configurații interzise. În faza de intensificare se pornește de la cea mai bună configurație descoperită, se eliberează lista tabu și se aplica din nou căutarea preliminară. În faza de diversificare se analizează configurațiile cel mai frecvent puse pe lista tabu, se selectează aleator una dintre acestea și se inițiază o căutare preliminară. Pentru anumite aplicații algoritmi TS s-au dovedit superiori algoritmilor de tip Simulated Annealing și celor evolutivi.

Cerinta: studiu bibliografic al algoritmilor de tip "tabu search" și implementarea unuia dintre algoritmi studiați (aplicat pentru o problemă de optimizare combinatorială sau de planificare).

Biblio. tehnici/TS

8. **VNS - Variable Neighbourhood Search (IACD, IASTE).** VNS este o metaeuristica recentă pentru rezolvarea problemelor de optimizare globală bazată pe definirea unui set de vecinătăți ale configurației curente (care se includ una în cealaltă) și construirea de noi configurații folosind aceste vecinătăți.

Cerința: Studiu bibliografic al algoritmilor de tip VNS și implementarea unui algoritm pentru rezolvarea unei probleme de optimizare la alegere.

Biblio: tehnici/VNS

9. **Hipereuristici - Hyperheuristics (IACD, IASTE)**. Sunt tehnici de generare a unor euristici destinate rezolvării unor probleme concrete. Euristică obținută trebuie să furnizeze pentru problema concretă de rezolvat soluții acceptabile în timp rezonabil. Generarea euristiciilor se poate baza pe algoritmi genetici (populația este în acest caz constituită din euristici care sunt transformate prin operatori specifici).

Cerința: Studiu bibliografic al hipereuristicilor și implementarea unei hipereuristici bazată pe algoritmi genetici (pentru rezolvarea unei probleme de optimizare combinatorială, de exemplu "bin-packing").

Biblio: tehnici/hyperheuristics

10. **Coevoluție cooperativă - Cooperative coevolution (IACD, IASTE)**. Se referă la evoluția în cooperare a mai multor populații. Furnizează o schemă generală de rezolvare a problemelor complexe prin divizarea lor în mai multe subprobleme și utilizarea câte unei populații pentru fiecare subproblemă. Strategia poate fi combinată cu orice tip de algoritm evolutiv și a fost aplicată cu succes în cazul problemelor de optimizare de dimensiuni mari.

Cerința. Studiul coevoluției, implementarea unui algoritm coevolutiv și testarea pentru o problemă de optimizare de dimensiuni mari (cel puțin 100).

Biblio. tehnici/coevolution

11. **Compact Evolutionary Algorithms (IACD, IS)**. Sunt algoritmi evolutivi destinați execuției pe dispozitive cu resurse limitate, de exemplu dispozitive mobile. Nu folosesc populații explicite ci descrieri ale distribuțiilor de probabilitate care modelează populația.

Cerința. Studiul unui algoritm evolutiv compact, implementarea și testarea pentru o problemă de optimizare.

Biblio. tehnici/compactEA

12. **KaizenProgramming (IASTE, IACD, IS)**. Metaeuristica "KaizenProgramming" (propusă în 2014) este inspirată de metoda de organizare a muncii în întreprinderi. Un task este dat unei echipe (manager și angajați) pentru a fi realizat într-o perioadă scurtă de timp pentru a evalua propriile noi soluții la o problemă.

Cerința. Studiul algoritmilor din această familie, implementarea și testarea unei variante pentru a rezolva o problemă de optimizare (la alegere dintre problemele test utilizate în literatură).

Biblio. tehnici/KaizenProgramming

13. **Cuckoo Search (IASTE, IACD)**. Metaeuristica "Cuckoo Search" (propusă în 2009) este inspirată de comportamentul cucilor de a folosi cuibul altor păsări pentru a-și depune ouăle. Ideea este modelată folosind mutații bazate pe distribuția Levy.

Cerința. Studiul algoritmilor din această familie, implementarea și testarea unei variante pentru a rezolva o problemă de optimizare (la alegere dintre problemele test utilizate în literatură).

Biblio. tehnici/CuckooSearch

14. **Firefly algorithms (IASTE, IACD)**. Metaeuristica "Firefly" este inspirată de modul de comunicare a licuricilor. Ideea este modelată prin mecanisme de atragere/respingere prin intermediul cărora este controlat procesul de căutare.

Cerința. Studiul algoritmilor din această familie, implementarea și testarea unei variante pentru a rezolva o problemă de optimizare (la alegere dintre problemele test utilizate în literatură).

Biblio. tehnici/FireflyAlgorithm

15. **Bat algorithms (IASTE, IACD).** Metaeuristica "Bat" este inspirată de mecanismul de eco-localitate folosit de lilieci pentru a se orienta. Ideea este modelată prin combinarea unei căutări locale cu una aleatoare.

Cerința. Studiul algoritmilor din această familie, implementarea și testarea unei variante pentru a rezolva o problemă de optimizare (la alegere dintre problemele test utilizate în literatură).

Biblio. tehnici/BatAlgorithm

16. **Search Based Software Engineering. (IS, IACD)** Proiectarea aplicațiilor bazată pe căutare se referă la utilizarea algoritmilor de căutare (în special metaeuristici) în: generarea automată de seturi de test, alocarea optimă a resurselor în cadrul unui proiect software, stabilirea etapelor de refactorizare, identificarea setului de cerințe care asigură compromisul dintre gradul de satisfacție a clienților și costurile de implementare etc.

Cerință: Studiu comparativ a diferitelor tehnici de proiectare software care implică tehnici de căutare bazate pe metaeuristici. Proiectul va include și implementarea unui exemplu.

Biblio. aplicatii/SearchBasedSoftwareEngineering

17. **Automated Program Repairing. (IS, IACD)** Generarea de secvențe scurte de program care pot fi utilizate pt a repara bug-uri poate fi făcută utilizând programare genetică. Generarea se bazează pe șabloane inițiale iar evaluarea comportării se face testând funcționarea programului pentru cazuri de test.

Cerință: Studiu comparativ a diferitelor abordări de "reparare" a programelor folosind programare genetică. Proiectul va include și implementarea unui exemplu.

Biblio. aplicatii/AutomatedProgramRepair

18. **Genetic Program Improvement. (IS, IACD)** Programarea genetică a fost recent utilizată în modificarea unor secvențe de cod (interpretate ca material genetic) cu scopul reducerii timpului de execuție sau a consumului de energie. În aceeași direcție s-au dezvoltat strategii de "transplant" a unor comportamente/funcționalități între diferite componente software (vezi <http://crest.cs.ucl.ac.uk/autotransplantation/>).

Cerință: Studiu comparativ a diferitelor abordări de eficientizare/adaptare a programelor folosind programare genetică. Proiectul va include și implementarea unui exemplu.

Biblio. aplicatii/GeneticImprovement

19. **Deep Learning** Tehnicile de învățare de tip "deep learning" s-au dovedit eficiente în recunoașterea scrisului de mână, a imaginilor și a vorbirii. Au fost aplicate cu succes în procesarea limbajului natural (de exemplu Google tool word2vec). Aceste tehnici se bazează pe modele de rețele neuronale cu arhitecturi complexe care conțin subrețele recurente (de tipul Boltzmann machines) și utilizează o pre-antrenare nesupervizată.

Cerința. Studiul modelelor de tip "deep neural networks" și a pachetelor Theano (<http://deeplearning.net/software/theano/>), PyLearn (<http://deeplearning.net/software/pylearn2/>) și word2vec (<https://code.google.com/p/word2vec/>).

Biblio. tehnici/DeepLearning

20. **Reservoir computing.**

Reservoir computing (<http://reservoir-computing.org/>, <http://reslab.elis.ugent.be/>) reprezintă o clasă de rețele neuronale recurente utilizate pentru prelucrarea seriilor temporale. Aceste modele conțin conexiuni ascunse având ponderi aleatoare iar singurele ponderi care sunt ajustate în procesul de antrenare sunt cele asociate conexiunilor către unitățile de ieșire.

Cerința. Studiul unui model (Echo State Network, Liquid State Machine etc) și implementarea unui exemplu simplu.

Biblio. tehnici/ReservoirComputing

21. **Implementarea pe GPU a algoritmilor evolutivi.** Implementarea algoritmilor evolutivi pe GPU a devenit populară în ultimii ani. Principalele aspecte ale unei astfel de implementări sunt legate de alegerea operațiilor care se implementează pe GPU și modul în care poate fi minimizat transferul între GPU și CPU.

Cerința: implementarea unui algoritm evolutiv simplu.

Biblio. aplicatii/EA+GPU

III. Teme orientate înspre implementare: analiza și testarea (prin implementarea unei aplicații) unor pachete software pentru algoritmi metaeuristici: IS, IACD, IASTE

Scopul acestor teme: studiul comparativ al unor biblioteci/pachete software destinate metaeuristicilor sau tematicilor înrudite. Necesită o echipă mai mare (atâtea persoane câte biblioteci/pachete vor fi analizate) și colaborarea între membrii echipei. La final se va prezenta pe lângă referatele individuale referitoare la fiecare bibliotecă/pachet un raport comun care să conțină rezultatele unui studiu comparativ.

Metaheuristics:

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. **MALLBA.** - library of C++ skeletons for combinatorial optimization <http://www.lsi.upc.edu/~mallba/>
8. **xxGA** - software library for parallel genetic algorithms. <http://neo.lcc.uma.es/software/xxga/index.php>
9. **EO.** <http://eodev.sourceforge.net/>
10. **ParadisEO.** <http://paradisEO.gforge.inria.fr/index.php?n=Main.HomePage>
11. **PISA.** <http://www.tik.ee.ethz.ch/sop/pisa/>
12. **OpenBeagle.** <https://github.com/chgagne/beagle>
13. **Robust Genetic Programming.** <http://robgp.sourceforge.net/about.php>

14. **DEAP - Distributed Evolutionary Algorithms in Python** <http://code.google.com/p/deap/>
15. **pySTEP - Python Strongly Typed gEnetic Programming** <http://pystep.sourceforge.net/>
16. **metaheuristic-algorithms-python 0.1.6** - Various metaheuristic algorithms implemented in Python <https://pypi.python.org/pypi/metaheuristic-algorithms-python/0.1.6>
17. **GPE - Genetic Programming Engine** <http://gpe.sourceforge.net/>
18. **MetsLib** - metaheuristic modeling framework and optimization toolkit in modern C++ <https://projects.coin-or.org/metslib>
19. **jmp75** - a simplified metaheuristics framework for .NET. <https://github.com/jmp75/metaheuristics>
20. **HEO** - cross-platform library that provides various parallel metaheuristic algorithms for hard optimization problems. <https://code.google.com/p/heo/>
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/>
23. **Jenes** - genetic algorithms for Java <https://sourceforge.net/projects/jenes/>
24. **JAMES** - A Java Metaheuristics Search Framework (discrete optimization using local search metaheuristics) <http://www.jamesframework.org/>
25. **oMetah** - a library aimed at the conception of metaheuristics (i.e. genetic/evolutionary algorithms, tabu search, simulated annealing, ant colony algorithms, etc.). <https://www.openhub.net/p/om>

Rețele neuronale:

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/>

Structura unui proiect. Fiecare proiect presupune:

- Redactarea unui referat privind tematica aleasă. Acesta va avea structura unui raport științific și va cuprinde: (a) un scurt rezumat (cca 5 rânduri) în care este enunțat scopul urmărit și principalul rezultat obținut; (b) o parte introductivă care cuprinde o sinteză privind stadiul actual în tematica studiată (bazată pe studiul bibliografic); (c) prezentarea detaliată a unuia sau mai multor modele/algoritmi/exemple ce fac obiectul studiului împreună cu opiniile/observațiile/soluțiile personale; (d) descrierea aplicației realizate cu comentarii privind modul de implementare, cu ilustrarea comportării pe probleme test etc.; (e) concluzii privind problematica studiată și eventuale direcții viitoare de studiu; (f) bibliografie (va conține toate materialele parcurse pentru realizarea referatului).

Dimensiunea referatului va fi cuprinsă între 6 și 12 de pagini. Referatul NU va fi o traducere a lucrărilor de la bibliografie ci o sinteză a acestora.

- Realizarea unei aplicații prin care să se illustreze specificul temei (în conformitate cu cerințele). Interesează în special implementarea metodei și mai puțin aspectele legate de interfață. Pentru testare se pot folosi probleme reale sau probleme sintetice.

Observații.

- Lucrările disponibile la <http://www.info.uvt.ro/~zflavia/AlgMeta/2017-2018/proiecte/> reprezintă puncte de pornire pentru proiect. Un proiect trebuie să se bazeze pe cel puțin două lucrări din listă.
- Raportul nu poate conține paragrafe întregi preluate din alte lucrări. Prezența unor astfel de paragrafe reprezintă plagiat și atrage direct respingerea proiectului.
- Proiectul va fi prezentat în timpul examenului. Pentru fiecare prezentare se alocă 10 minute și presupune pregătirea unui set de 10-12 slide-uri.