Teme pentru proiecte - Algoritmi metaeuristici

2016-2017

I. Teme orientate înspre aplicaţii

Specificul acestor teme este faptul că se porneşte de la o problemă concretă si trebuie identificată şi implementată o metodă de rezolvare. Metodele vizate se bazează pe diferite tipuri de meta-euristici (inclusiv algoritmi evolutivi si alte meta-euristici inspirate de natură) precum şi modele bazate pe învăţare automată, cum sunt reţelele neuronale.

1. **Alocarea resurselor în cloud - Cloud Resource Provisioning (IACD, IS).** Se consideră problema asignării de resurse cloud pentru o aplicaţie specifică astfel încât să fie satisfăcute cerinţe privind caracteristicile resurselor de calcul şi restricţii privind costul. Problema poate fi abordată din cel puţin două perspective: cea a utilizatorului (conducând la problema selecţiei maşinilor virtuale) şi cea a provider-ului (conducând la problema mapării maşinilor virtuale pe maşini fizice). In ambele situatii se poate ajunge la rezolvarea unor probleme de optimizare care implică restricţii şi/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 şi simularea unei infrastructuri de tip cloud oferind suport pentru simularea de maşini virtuale, centre de date, diferite politici de alocare a resurselor precum si controlul consumul de energie.

   Cerinţa: Selecţia unei probleme şi a unei metode, implementarea metodei şi testarea folosind CloudSim.

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

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

2. **Optimizarea traseelor - Vehicle Routing Problem (IACD, IASTE).** VRP (Vehicle Routing Problem) este o generalizare a problemei comis voiajorului şi are ca scop optimizarea traseelor mai multor mijloace de transport care asigură aprovizionarea mai multor clienţi cu produse preluate de la unul sau mai multe depozite. Există numeroase variante ale problemei care diferă între ele prin restricţiile impuse (ex: capacitatea limitată a vehiculelor, intervalele de timp în care trebuie realizată aprovizionarea, modificarea cerinţelor în mod dinamic etc) sau prin criteriul/criteriile de optim (cost, distanţă parcursă, număr clienţi serviţi 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 selecţie, alocare, împachetare, ordonare şi poate fi rezolvată utilizând diferite tipuri de euristici şi metaeuristici.

   Cerinţa: Selecţia unei probleme şi a unei metode, implementarea metodei şi testarea pentru una din instanţele de la http://neo.lcc.uma.es/vrp/

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

Biblio: aplicatii/VehicleRouting

3. **Planificarea turelor intr-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 restricţiilor 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.

   Cerinţa: Selectia unei metode adecvate (EA, SA, ACO, sau altă metodă), prezentarea metodei, implementarea si testarea pentru o instanţă a problemei.

   Biblio: aplicatii/nurse-rostering

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

4. **Generarea orarelor - Timetabling (IACD, IASTE).** Planificarea unei activităţi (de exemplu construirea unui orar) este o problemă cu numeroase restricţii şi unul sau mai multe criterii de optimizat. Unele restricţii sunt puternice (trebuie respectate întotdeauna) iar altele sunt mai slabe (pot fi încălcate însă induc o creştere 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).

   Cerinţa: Selecţia unei metode adecvate (TS, SA, GA sau altele), prezentarea metodei, implementarea şi testarea pentru o instanţă a problemei (generarea unui orar).

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

   Biblio: aplicatii/timetabling

5. **Optimizarea semaforizării - Traffic Lights Optimization (IS, IACD, IASTE).** Se referă la stabilirea duratei pentru fiecare dintre stările unui semafor (roşu, verde) astfel încât să fie optimizate anumite criterii (de exemplu să fie maximizat numărul vehiculelor care ajung la destinaţie şi/sau minimizarea timpului de deplasare per vehicul între două puncte). In contextul utilizării de metaeuristici bazate pe populaţii, o soluţie este reprezentată ca un vector de valori care reprezintă durata în secunde corespunzătoare celor două stări (roşu şi verde) corespunzătoare tuturor semafoarelor din zona analizată. Evaluarea calităţii unei soluţii se bazează pe simularea traficului

   Cerinţa: Studiul unui model de optimizare bazat pe metaeuristici (vezi lucrările de la bibliografie), implementarea unei variante şi testarea pe un scenariu simplu (de exemplu 1-2 intersecţii din Timişoara) 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

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

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

Biblio: aplicatii/RegularExpressionsInference

7. **Optimizarea portofoliului - Portfolio optimization (IACD, IASTE).** Presupune construirea portofoliilor de investiţii astfel încât să fie maximizat profitul şi minimizate riscurile. Problema poate fi formulată ca o problemă de optimizare combinatorială ce poate fi rezolvată utilizând: algoritmi genetici (GA), programare genetică (GP), particle swarm optimization (PSO), Hopfield Neural Networks (HNN)

   Cerinţa: Identificarea şi descrierea unei metode (bazate pe GA, GP, PSO, HNN), implementarea şi testarea ei.

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

   Biblio: aplicatii/portfolio_optimization

8. **Selecţia 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) şi de tip wrapper (încorporat în procesul de clasificare). Problema de selecţie poate fi formulată ca o problemă de optimizare uni sau multi-criterială şi poate fi rezolvată folosind algoritmi evolutivi (EA), ant colony optimization (ACO), tabu search (TS) etc.

   Cerinţa: Implementarea unui algoritm pentru selecţia atributelor (de tip filtru sau wrapper combinat fie cu algoritmi de clasificare fie cu algoritmi de grupare) şi testarea pentru date de la UCI Machine Learning repository (http://www.ics.uci.edu /~mlearn /MLRepository.html).

   Variante: 3 (EA, ACO, TS)

   Biblio: aplicatii/feature_selection

9. **Detectarea comunităţilor in reţele sociale - Community detection (IACD, IS, IASTE).** Are ca scop identificarea grupurilor de elemente puternic interconectate (de exemplu utilizatori ai unei reţele sociale). Este echivalentă cu problema partiţionarii grafurilor în submulţimi de noduri astfel încât nodurile din aceeaşi submulţime au grad mare de interconectare pe când nodurile din submulţimi diferite au grad scăzut de interconectivitate. Problema se poate rezolva aplicând tehnici de grupare (partiţionale, ierarhice sau spectrale) dar şi utilizând algoritmi evolutivi.

   Cerinţa: Studiul problemei detectării comunităţilor, implementarea unui algoritm evolutiv (dintre cei descrişi în lucrările de la bibliografie) si testarea pentru o problemă de test (de exemplu problema "Zacharys karate club network").

   Biblio: aplicatii/communities_detection

10. **Sisteme de detecţie a intruziunilor - Intrusion detection systems (IACD, IS).** Sistemele de detectare a intruşilor se bazează pe clasificarea comportării unui "utilizator" al unui sistem de calcul în normală sau anormală folosind date obţinute prin monitorizarea evenimentelor ce apar în cadrul sistemului. Există diferite soluţii bazate pe tehnici din calculul inteligent: retele neuronale (NN), algoritmi evolutivi (EA) şi sisteme imunitare artificiale (AIS). Produsele antivirus actuale (de exemplu Bitdefender) încorporează algoritmi de învăţare automată (de exemplu clasificatori de tip perceptron) sau algoritmi evolutivi pentru extragerea atributelor relevante pentru detecţie.

    Cerinţa: Analiza structurii unui IDS şi implementarea unei componente ce foloseşte una dintre tehnicile: NN, EA or AIS.

Variante: 3 (NN, EA, AIS)

Biblio: aplicatii/intrusion_detection_systems

11. **Testarea aplicaţiilor software - Evolutionary software testing (IS, IACD)** Algoritmii evolutivi sau metaeuristicile inspirate de natura (de exemplu ACO) au fost recent utilizaţi în generarea automată a cazurilor de test pentru testarea structurală a produselor software.

    Cerinţa: Studiu bibliografic al utilizării algoritmilor evolutivi in testarea produselor software şi implementarea unui algoritm pentru generarea datelor de test pentru o aplicaţie la alegere.

    Biblio. aplicatii/software_testing

12. **Evolutionary Art. (IASTE, IACD)** Algoritmii metaeuristici pot fi utilizati pentru a sintetiza imagini artistice pornind der la şabloane şi aplicând algoritmi de încrucişare şi mutaţie. In general evaluarea calităîi rezultatelor se bazează pe interacţiunea cu utilizatorul care poate acorda scoruri configuraţiilor intermediare.

    Cerinţa: Implementarea unui algoritm metaeuristic pentru sinteza unor imagini artistice sau a unor compoziţii muzicale (la alegere).

    Biblio: aplicatii/EvolutionaryArt

13. **Segmentarea imaginilor - Multilevel Thresholding** Segmentarea imaginilor presupune identificarea de regiuni în imagini astfel încât pixelii din aceeaşi 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 mai mai multe valori prag (multilevel thresholding) care se bazează pe ideea de a determina valori ale pragurilor care maximizează varianţa sau entropia calculată la nivelul regiunilor.

    Cerinţa: 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 şi predicţie.** În ultimii ani, în cadrul conferinţelor majore din domeniul calculului evolutiv (GECCO, CEC) au fost organizate competiţii de soluţii la probleme provenite din industrie.

    Cerinţa: Implementarea unei soluţii la una dintre probleme enunţate 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 particularităţile şi aplicabilitatea unor tehnici metaeuristice. Proiectul presupune un studiu bibliografic al tehnicii şi implementarea unui exemplu.

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

   Biblio. tehnici/ABC

2. **AS - Ant systems (IACD, IASTE).** Comportarea insectelor ce trăiesc în colectivităţi (furnicile şi albinele) reprezintă sursă de inspiraţie pentru dezvoltarea unor metode de optimizare (pentru rezolvarea unor probleme dificile de optimizare combinatorială - problema comis voiajorului, planificarea activităţilor etc.) şi control distribuit (de exemplu rutarea adaptivă în sistemele de comunicaţii). Abilitatea acestor comunităţi de a rezolva probleme se bazează pe comunicarea pe cale chimică (prin intermediul feromonilor): pe traseul parcurs pentru căutarea hranei furnicile lasă o "urmă" care permite identificarea ulterioară a traseului care a dus la găsirea hranei. Sistemele artificiale implementează acţiunea feromonilor prin "recompensarea" configuraţiilor bune şi evaporarea feromonilor prin "uitarea" unor configuraţii.

   Cerinţa. Studiu bibliografic al sistemelor de tipul coloniilor de furnici şi implementarea unui algoritm din această clasă pentru rezolvarea unei probleme de optimizare (variante: (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 distribuţia geografică a organismelor biologice. Utilizează operatori speciali de încrucişare inspiraţi de procesele de emigrare şi imigrare.

   Cerinţa. Studiul BBO, implementarea unei variante şi 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 inspiraţie comportarea socială (legată în special de transmiterea şi utilizarea în comun a informaţiei) a unor fiinţe vii cum sunt stolurile de păsări, roiurile de albine sau bancurile de peşti. În variantele artificiale procesul de căutare este asigurat de un ansamblu de "particule" a căror mişcare este caracterizată de o "viteză" care se modifică în timp în funcţie caracteristicile întregului sistem. Permit găsirea rapidă a optimului însă au dificultăţi în evitarea minimelor locale. Recent au fost extinse şi pentru rezolvarea problemelor de optimizare multicriterială şi pentru optimizarea dinamică.

   Cerinţa: Studiu bibliografic a algoritmilor de tip PSO şi implementarea unui algoritm de tip PSO pentru rezolvarea unei probleme de optimizare continuă (funcţie 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 apariţia unui agent patogen reacţionează 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 aceluiaşi agent sistemul să reacţioneze rapid). Sistemele imunitare artificiale sunt utilizate pentru rezolvarea unor probleme de clasificare (pe baza proprietăţii de afinitate a unei celule imunitare pentru un anumit agent), pentru optimizare şi pentru detectarea atacurilor asupra calculatoarelor (de tipul viruşilor).

   Cerinţa: Studiu bibliografic al sistemelor imunitare artificiale şi 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 distributiilor de probabilitate corespunzatoare configuratiilor de succes din populatia curenta. S-au aplicat cu succes

   Cerinţa: Studiu bibliografic al algoritmilor de tip EDA şi 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 configuraţii prohibite. A fost propusă de Glover in 1986 şi a fost aplicată cu succes în rezolvarea problemelor de optimizare cu restricţii (de exemplu in probleme de planificare a activităţilor). Tabu search consta în trei faze: căutare preliminară, intensificare şi diversificare. Prima etapă consta in selecţia din vecinătatea unei configuraţii curente a celei mai bune configuraţii din perspectiva funcţiei obiectiv. Noua configuraţie este acceptată chiar dacă este mai rea decât cea curentă. Pentru a evita trecerea succesivă între două configuratii (intrarea într-un ciclu) se gestionează o listă de configuraţii interzise. In faza de intensificare se porneşte de la cea mai bună configuraţie descoperită, se eliberează lista tabu şi se aplica din nou căutarea preliminară. In faza de diversificare se analizează configuraţiile cel mai frecvent puse pe lista tabu, se selectează aleator una dintre acestea şi se initiaa o căutare preliminară. Pentru anumite aplicaţii algoritmii TS s-au dovedit superiori algoritmilor de tip Simulated Annealing si celor evolutivi.

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

   Biblio. tehnici/TS

8. **VNS - Variable Neighbourhood Search (IACD, IASTE).** VNS este o metaeuristica recenta pentru rezolvarea problemelor de optimizare globala bazata pe definirea unui set de vecinătăţi ale configuraţiei curente (care se includ una în cealaltă) şi construirea de noi configuraţii folosind aceste vecinătăţi.

   Cerinţa: Studiu bibliografic al algoritmilor de tip VNS şi 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 rezolvarii unor probleme concrete. Euristica obtinuta trebuie sa furnizeze pentru problema concreta de rezolvat soluţii acceptabile în timp rezonabil. Generarea euristicilor se poate baza pe algoritmi genetici (populatia este in acest caz constituita din euristici care sunt transformate prin operatori specifici).

   Cerinţa: Studiu bibliografic al hipereuristicilor şi implementarea unei hipereuristici bazata pe algoritmi genetici (pentru rezolvarea unei probleme de optimizare combinatoriala, de exzemplu "bin-packing").

   Biblio: tehnici/hyperheuristics

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

    Cerinţa. Studiul coevoluţiei, implementarea unui algoritm coevolutiv şi testarea pentru o problemă de optimizare de dimensiuni mari (cel puţin 100).

    Biblio. tehnici/coevolution

11. **Compact Evolutionary Algorithms (IACD, IS).** Sunt algoritmi evolutivi destinaţi execuţiei pe dispozitive cu resurse limitate, de exemplu dispozitive mobile. Nu folosesc populaţii explicite ci descrieri ale distribuţiilor de probabilitate care modelează populaţia.

    Cerinţa. Studiul unui algoritm evolutiv compact, implementarea şi testarea pentru o problemă de optimizare.

    Biblio. tehnici/compactEA

12. **Cuckoo Search (IASTE, IACD).** Metaeuristica "Cuckoo Search" (propusă în 2009) este inspirată de comportamentul cucilor de a folosi cuibul altor păsări pentru a-şi depunde ouăle. Ideea este modelată folosind mutaţii bazate pe distribuţia Levy.

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

    Biblio. tehnici/CuckooSearch

13. **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.

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

    Biblio. tehnici/FireflyAlgorithm

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

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

Biblio. tehnici/BatAlgorithm

15. **Search Based Software Engineering. (IS, IACD)** Proiectarea aplicaţiilor bazată pe căutare se referă la utilizarea algoritmilor de căutare (în special metaeuristici) în: generarea automată de seturi de test, alocarea optimală a resurselor în cadrul unui proiect software, stabilirea etapelor de refactorizare, identificarea setului de cerinţe care asigură compromisul dintre gradul de satisfacţie a clienţilor şi costurile de implementare etc.

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

    Biblio. aplicatii/SearchBasedSoftwareEngineering

16. **Automated Program Repairing. (IS, IACD)** Generarea de secvenţe scurte de program care pot fi utilizate pt a repara bug-uri poate fi făcută utilizând programare genetică. Generarea se bazează pe şabloane iniîale iar evaluarea comportării se face testâand funcţionarea programului pentru cazuri de test.

    Cerină: Studiu comparativ a diferitelor abordări de "reparare" a programelor flosind programare genetică. Proiectul va include şi implementarea unui exemplu.

    Biblio. aplicatii/AutomatedProgramRepair

17. **Genetic Program Improvement. (IS, IACD)** Programarea genetică a fost recent utilizată în modificarea unor secvenţe de cod (interpretate ca material genetic) cu scopul reducerii timpului de execuţie sau a consumului de energie. In aceeaşi direcţie s-au dezvoltat strategii de "transplant" a unor comportamente/funcţionalităţi între diferite componente software (vezi http://crest.cs.ucl.ac.uk/autotransplantation/).

    Cerină: Studiu comparativ a diferitelor abordări de eficientizare/adaptare a programelor flosind programare genetică. Proiectul va include şi implementarea unui exemplu.

    Biblio. aplicatii/GeneticImprovement

18. **Deep Learning** Tehnicile de învăţare de tip "deep learning" s-au dovedit eficiente în recunoaşterea scrisului de mână, a imaginilor şi a vorbirii. Au fost aplicate cu succes în procesarea limbajului natural (de exemplu Google tool word2vec). Aceste tehnici se bazează pe modele de reţele neuronale cu arhitecturi compleze care conţin subreţele recurente (de tipul Boltzmann machines) şi utilizează o pre-antrenare nesupervizată.

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

    Biblio. tehnici/DeepLearning

19. **Reservoir computing.**
    Reservoir computing (http://reservoir-computing.org/, http://reslab.elis.ugent.be/) reprezintă o clasă de reţele neuronale recurente utilizate pentru prelucrarea seriilor temporale. Aceste modele conţin conexiuni ascunse având ponderi aleatoare iar singurele ponderi care sunt ajustate în procesul de antrenare sunt cele asociate conexiunilor către unităţile de ieşire.

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

Biblio. tehnici/ReservoirComputing

20. **Implementarea pe GPU a algoritmilor evolutivi.** Implementarea algoritmilor evolutivi pe GPU a devenit populară în ultimii ani. Principalale aspecte ale unei astfel de implementări sunt legate de alegerea operaţiilor care se implementeaza pe GPU şi modul în care poate fi minimizat transferul între GPU şi CPU.

    Cerinţa: implementarea unui algoritm evolutiv simplu.

    Biblio. aplicatii/EA+GPU

III. Teme orientate înspre implementare: analiza si testarea (prin implementarea unei aplicatii) unor pachete software pentru algoritmi metaeuristic: IS, IACD, IASTE

Scopul acestor teme: studiul comparativ al unor biblioteci/pachete software destinate metaeuristicilor sau tematicilor înrudite. Necesită o echipa mai mare (atâtea persoane cate biblioteci/ pachete vor fi analizate) şi colaborarea între membrii echipei. La final se va prezenta pe langa referatele individuale referitoare la fiecare bibliotecă/pachet un raport comun care să conţina 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://neo.lcc.uma.es/software/ma
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. **PerlGP** - Perl Genetic Programming System http://perlgp.org/
19. **MetsLib** - metaheuristic modeling framework and optimization toolkit in modern C++ https://projects.coin-or.org/metslib
20. **jmp75** - a simplified metaheuristics framework for .NET. https://github.com/jmp75/metaheuristics
21. **HEO** - cross-platform library that provides various parallel metaheuristic algorithms for hard optimization problems. https://code.google.com/p/heo/
22. **OPT4J** - open source Java-based framework for evolutionary computation. http://opt4j.sourceforge.net/
23. **HeuristicLab** - a framework for heuristic and evolutionary algorithms. http://dev.heuristiclab.com/
24. **Jenes** - genetic algorithms for Java https://sourceforge.net/projects/jenes/
25. **JAMES** - A Java Metaheuristics Search Framework (discrete optimization using local search metaheuristics) http://www.jamesframework.org/
26. **oMetah** - a library aimed at the conception of metaheuristics (i.e. genetic/evolutionnary algorithms, tabu search, simulated annealing, ant colony algorithms, etc.). https://www.openhub.net/p/om

Reţele 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 ştiinţific şi va cuprinde: (a) un scurt rezumat (cca 5 rânduri)în care este enunţat scopul urmărit şi principalul rezultat obţinut; (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/observaţiile/soluţiile personale; (d) descrierea aplicaţiei realizate cu comentarii privind modul de implementare, cu ilustrarea comportării pe probleme test etc.; (e) concluzii privind problematica studiată şi eventuale direcţii viitoare de studiu; (f) bibliografie (va conţine toate materialele parcurse pentru realizarea referatului).

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

- Realizarea unei aplicaţii prin care să se ilustreze specificul temei (în conformitate cu cerinţele). Interesează în special implementarea metodei şi mai puţin aspectele legate de interfaţă. Pentru testare se pot folosi probleme reale sau probleme sintetice.

Observaţii.

- Lucrările disponibile la http://www.info.uvt.ro/~dzaharie/am2016/proiecte
  reprezintă puncte de pornire pentru proiect. Un proiect trebuie să se bazeze pe cel puţin două lucrări din listă.

- Raportul nu poate conţine paragrafe întregi preluate din alte lucrări. Prezenţa unor astfel de paragrafe reprezintă plagiat şi atrage direct respingerea proiectului.

- Proiectul va fi prezentat în timpul examenului. Pentru fiecare prezentare se alocă 10 minute şi presupune pregătirea unui set de 10-12 slide-uri.