Anca MIRON

# Aplicații ale IA în managementul energiei

Suport de curs

U.T.PRESS
Cluj-Napoca, 2025
ISBN 978-606-737-790-3

**Anca MIRON**

# Aplicații ale IA în managementul energiei

*Suport de curs*

**U.T.PRESS**
**Cluj-Napoca, 2025**
**ISBN 978-606-737-790-3**

Editura U.T.PRESS
Str. Observatorului nr. 34
400775 Cluj-Napoca
Tel.: 0264-401.999
e-mail: utpress@biblio.utcluj.ro
www.utcluj.ro/editura

Recenzia: Prof.dr.ing. Sorin G. Pavel
Conf.dr.ing. Andrei C. Cziker

Pregătire format electronic on-line: Gabriela Groza



**ISBN 978-606-737-790-3**

# PREFAŢĂ

Prezentul material didactic este destinat să ofere un suport bibliogafic pentru informațiile predate și discutate în cadrul activității de curs din programa analitică a disciplinei *Aplicații ale inteligenței artificiale în managementul energiei,* pentru anul I, studii de master, specializarea de energetică.

Suportul de curs a fost conceput ca un material didactic complementar orelor de curs, astfel încât noţiunile teoretice și practice acumulate să confere studenţilor un bagaj de cunoştinţe menit să le permită abordarea cu succes a problemelor pe care vor trebui să le rezolve în calitate de specialişti energeticieni.

Elaborate în această concepţie, toate capitolele acestui material didactic sunt logic structurate, fiecare fiind corespondent unui curs și beneficiază de o tratare teoretică corespunzătoare. De asemenea, s-a urmărit ca fiecare subiect să asigure și exemple cuprinzătoare, menite să exemplifice principalele aspecte ale subiectelor studiate.

Cu toate eforturile depuse, autoarea este convinsă că această formă a prezentului material didactic poate fi îmbunătăţită prin continuarea preocupărilor în domeniu, prin sugestiile studenţilor care îl vor utiliza precum și ale altor specialişti.

Autoarea

# CUPRINS

# Introducere

Inteligenţa Artificială, IA este domeniul ştiinţific dedicat realizării de sisteme care să funcționeze și să rezolve probleme asemenea oamenilor. În [1] se specifică următoarele: „Din punctul de vedere al unui inginer, Inteligenţa Artificială urmăreşte generarea unor reprezentări şi proceduri concepute pentru rezolvarea automată a problemelor care până atunci au fost rezolvate de oameni. Scopul Inteligenței Artificiale este înţelegerea ca o formă de manifestare a calculelor".

Realizările din domeniul inteligenței artificiale așa cum sunt astăzi se datorează dezvoltării ştiinţei calculatoarelor, aprofundării înţelegerii modului de funcţionare a creierului uman și a comportamentului uman. Intr-adevăr, în ultimele trei decenii se observă o explozie a aplicațiilor tehnicilor de IA în majoritatea domeniilor de activitate umană: inginerie, medicină, economie, tehnică, industria armamentului etc.

Tehnicile de IA reprezintă modalităţile de implementare a principiilor Inteligenței Artificiale pentru soluţionarea diferitelor probleme abordate. Conform spuselor multor specialişti, tehnicile de IA sunt metodele de rezolvare ale secolului XXI [2].

Inteligenţa Artificială a interesat omul încă din antichitate, când a apărut prima dată ideea de robot și mecanism care se mișcă singur. Insă, punctul de start al științei cu același nume este abia anul 1943 când apare primul model de neuron formal, propus de neuro-fiziologul W.S. McCulloch și matematicianul W. Pitts.

În aceeași perioadă, în S.U.A. şi Germania au apărut primele calculatoare electronice, care au făcut posibil progresul și dezvoltarea ulterioară a tehnicilor de IA [2]. Câțiva ani mai târziu, în 1949 a apărut primul program comercial (software) care putea fi asimilat de un calculator electronic. Această invenţie oferea noi perspective în domeniul calculatoarelor și implicit al IA. Apoi, la sfârşitul anului 1956, a fost dezvoltat primul program de IA, *The Logic Theorist*, de către A.Newell, C.Shaw și H.A.Simon. Acest software intra în categoria programelor de rezolvare a problemelor generale (general-problem solver). În același an, John McCarthy, considerat părintele IA, a organizat o conferinţă intitulată „The Dartmouth Summer Research Project on Artificial Intelligence". În cadrul acestei manifestări ştiinţifice a fost utilizat pentru prima dată termenul

de Inteligenţă Artificială (Artificial Inteligence), şi au fost discutate și conturate direcţiile de cercetare din domeniu inteligenței artificiale.

> Specialiști din domeniu au spus atunci: scopul Inteligenței Artificiale este de a realiza sisteme care: gândesc ca oamenii, acţionează ca oamenii, gândesc raţional și acţionează raţional.

Aceste scopuri au deschis două direcţii fundamentale de cercetare: procese cognitive (gândire și raţionament) şi comportament.

Direcția proceselor cognitive cuprinde următoarele tehnici de IA:

1. Învățare automată (ML): algoritmi care permit sistemelor să învețe din date și să se îmbunătățească în timp fără programare explicită. Exemplele includ învățarea supravegheată, nesupravegheată și de întărire.
2. Învățare profundă (adâncă): un subset de ML care utilizează rețele neuronale cu mai multe straturi pentru a învăța modele complexe din seturi mari de date, utile în sarcini precum recunoașterea imaginilor și a vorbirii.
3. Procesarea limbajului natural (NLP – Natural Language Processing): tehnici care permit mașinilor să înțeleagă, să interpreteze și să genereze limbajul uman, permițând aplicații precum chatbot-uri și traducerea limbii naturale.
4. Reprezentarea și raționamentul cunoștințelor: metode de reprezentare a informațiilor despre lume într-o formă pe care un sistem informatic o poate utiliza pentru a rezolva sarcini complexe, cum ar fi luarea deciziilor și inferența.
5. Arhitecturi cognitive: cadre concepute pentru a imita procesele cognitive umane, cum ar fi ACT-R și SOAR, care modelează memoria, atenția și rezolvarea problemelor.

Comportamentul uman este simulat prin intermediul tehnicilor de IA:

1. Sisteme experte: sisteme IA care emulează capacitatea de luare a deciziilor a unui expert uman folosind o bază de cunoștințe și reguli de inferență.
2. Logica fuzzy: o formă de logică care se ocupă mai degrabă de raționamentul aproximativ decât de soluții fixe și exacte, utilă în sistemele care necesită luare a deciziilor de tip uman.

3. Algoritmi evolutivi: tehnici de optimizare inspirate din selecția naturală, cum ar fi algoritmii genetici și programarea genetică, utile în rezolvarea adaptivă a problemelor.
4. Modelare bazată pe agenți: simulează interacțiunile agenților autonomi pentru a evalua efectele acestora asupra sistemului în ansamblu; Este utilizată pe scară largă pentru modelarea comportamentului social și simulările complexe ale sistemelor.
5. Emotion AI (Affective Computing): recunoaște și simulează emoțiile umane folosind viziunea computerizată, NLP și analiza vorbirii pentru a face IA mai empatică și mai interactivă.
6. Clonarea comportamentală: un tip de învățare supravegheată în care modelele de IA învață din demonstrații umane pentru a imita acțiuni, adesea folosite în robotică și jocuri.

In continuare, acest suport de curs are următoarea structură:

- Abrevieri – cuprinde lista tuturor abrevierilor folosite în corpul cărții;
- Aspecte generale ale inteligenței artificiale - este capitolul care introduce termenii de bază ale tehnicilor IA din prezenta carte. De asemenea, se prezintă și o scurta istorie a tehnicilor de IA.
- Logica fuzzy, mulțimi fuzzy – prezintă principiile de bază ale logicii fuzzy.
- Numere fuzzy – se focalizează pe caracteristicile numerelor fuzzy care sunt folosite în aplicațiile caracteristice managementului energiei.
- Sisteme fuzzy – descrie caracteristicile sistemelor fuzzy care sunt folosite în managementul energiei.
- Aplicații ale logicii fuzzy – prezintă aplicații din literatura de specialitate.
- Rețele neuronale artificiale, RNA – face o introducere în tematica rețelelelor neuronale artificiale.
- Arhitecturi neuronale clasice – se focalizează pe RNA cu arhitectură clasică.
- Arhitecturi neuronale avansate – prezintă rețelele neuronale avansate, care stau la baza tuturor aplicațiilor RNA care se întâlnesc în prezent în societatea umană.
- Aplicații ale RNA – descrie aplicații ale RNA în managementul energiei diseminate în literatura de specialitate.

- Algoritmi genetici, AG – face o introducere în tematica algoritmilor genetici.
- Operatorii genetici - descrie modul în care sunt folosiți principalii operatori genetici, care stau la baza funcționalității AG.
- Populații și generații – prezintă caracteristicile populațiilor și generațiilor cu care lucrează AG.
- Aplicații ale AG – cuprinde descrierea unor aplicații ale AG în managmentul energiei culese din literatura de specialitate.
- Concluzii – se face o trecere în revistă a principalelor aspecte prezentate în curs, și se trag concluziile necesare.
- Anexe – se prezintă câte un exemplu numeric pentru cele trei tehnici de IA prezentate în cadrul cursului.

# Abrevieri

ABC – Artificial bee colony, colonie de albine artificială
ACO – Ants colony optimization, optimizarea coloniei de furnici
AG – Algoritmi genetici
ANFIS – Adaptive Neuro-Fuzzy Interface System
BERT – Bidirectional encoder representations from Transformers (IA)
CNN – Convolutional neural networks
DE – Differential evolution, evoluție diferențială
DL – Deep Learning, învățare adâncă
DQN – Deep Q-Network
FDL – Fuzzy Logic Designer, MATLAB
FLT – Fuzzy Logic Toolkit, LABVIEW
GAN – Generative adversarial network
GPT – Generative Pre-trained transformer
HIL – hardware-in-the-loop
HMM – hidden Markov model, modelul Markov ascuns
HVAC – Heating Ventilation Aer-Conditioning
IA – Inteligență Artificială
IBM – International Business Machines Corporation
k-NN – k-nearest neighbor algorithm
LF – Logica fuzzy
LISP – LISt Programming
LSTM – Long Short-Term memory
MDP – Markov decision process
MIT – Massachusetts Institute of Technology
ML – Machine Learning
MLP – Multistrat Perceptron – perceptrol multistrat
MPPT – maximum power point track
PCA – Principal component analysis
PROLOG –PROgramming in LOGic
PSO – Particle swarm optimization
RL – Reinforcement learning
RNA – Rețele neuronale artificiale
RNN – Recurrent neural network (Rețele neuronale recurente)
SE – Sisteme Expert
SGF – sisteme genetic fuzzy

SVM – Support vector machine
VHDL - Very Mare Speed Integrated Circuit Hardware Description Language

# 1

# Aspecte generale ale Inteligenței Artificiale

Acest capitol prezintă aspectele de bază ale tehnicilor de IA cuprinse în acest curs. De asemenea, se face o introducere în IA, precum și o scurtă prezentare a istoriei tehnicilor de IA studiate. Capitolul se încheie cu un subcapitol care enumera domeniile de aplicare ale IA.

## 1.1 Inteligenta Artificială

Termenul de Inteligenţă Artificială (IA) a fost introdus în anul 1956 în cadrul primei conferințe pe tema tehnicilor de inteligență artificială.

Inteligenţa artificială ascunde mai multe aspecte ale modului de a fi uman: a gândi ca un om, a gândi rațional, a acţiona ca un om, și a acționa rațional.

***A gândi ca un om***

A gândi ca un om – oamenii de știință care au pus bazele IA s-au referit la realizarea unor calculatoare care gândesc, maşini cu minţi, în adevăratul sens al cuvântului. În acest sens, matematicianul și specialistul în Computer Science (știința claculatoarelor) Richard Bellman a vorbit despre automatizarea unor activităţi caracteristice oamenilor, precum decizia, rezolvarea problemelor, învăţarea etc. El a dezvoltat programarea dinamică, o tehnică fundamentală folosită ulterior în optimizare și luarea deciziilor. Astfel, R. Bellman a contribuit semnificativ la IA și teoria controlului, punând bazele necesare dezvoltării ulterioare a proceselor de decizie Markov, care sunt folosite intensiv în IA, în special în învățarea întărită (reinforcement learning).

Această abordare se referă la modelarea cognitivă, la modul cum gândesc oamenii. Modelarea cognitivă se poate realiza prin scanarea creierului, introspecţie și experimente psihologice. Astfel, se observă că modelarea cognitivă se implică atât partea psihologică a oamenilor, cât și fiziologică a

creierului uman. Odată determinat modul de gândire uman, și realizarea unui algoritm, modelul poate fi implementat sub forma unui program informatic.

Se poate deduce din cele precizate anterior că, dezvoltarea modelelor cognitive nu era posibilă dacă dezvoltarea tehnologică și înțelegerea creierului uman nu era la nivelul necesar.

***A gândi raţional***

Filosoful grec Aristotel a fost primul care a încercat să codifice "gândirea corectă". El este cosiderat părintele logicii, fiind cel care a pus bazele raționamentului deductiv, respectiv inductiv. Ulterior studiile lui și a celor care i-au urmat au dus la crearea unui nou domeniu științific – logica.

Mult mai târziu, în secolul XIX s-a dezvoltat un cod privind legăturile logice dintre diferite obiecte. Aceste legături logice au fost baza algebrei booleene, unde cei mai uzuali operatorii logici sunt ȘI, SAU etc. În momentul de față, algebra booleană este aplicată în circuitele electronice, comunicații și IA.

Dezavantaje utilizării logicii fac referire la două aspecte importante, care de multe ori nu pot fi neglijate:

- Operaţiile logice necesare pentru rezolvarea unei probleme complicate pot epuiza dispozitivele de calcul.
- Dificultatea codificării în termeni logici ai informaţiei incomplete.

***A acţiona ca un om***

A acționa ca un om se referă la realizarea de maşini care execută acţiuni ce necesită inteligenţă asemănătoare ființelor umane. Specialiști în IA au făcut referire la comportamentul inteligent al dispozitivelor.

În 1950 Alan Turing a dezvoltat un test, denumit ulterior testul Turing, prin care se testează dacă un dispozitiv acționează ca un om. Acest test, încă este folosit ca etalon în analiza IA.

Testul Turing constă în parcurgerea următoarelor etape:

1. Un specialist uman interacționează atât cu un om, cât și cu o mașină prin intermediul comunicării bazate pe text (pentru a preveni părtinirea bazată pe aspect sau voce).
2. Specialistul pune întrebări și analizează răspunsurile.
3. Dacă specialistul nu poate spune în mod sigur care este mașina, se consideră că mașina bazată pe IA a trecut testul.

Când a construit acest test, Turing nu a răspuns direct la întrebarea

„Pot mașinile să gândească?”
ci a reformulat întrebarea în una mai practică
„Poate o mașină să comunice într-un mod care nu se poate distinge de un om?”

Limitările și criticile aduse la adresa testului Turing sunt:

- Măsoară înșelăciunea, nu inteligența adevărată (un chatbot ar putea trece prin imitarea oamenilor fără a înțelege adevăratul sens al întrebărilor).
- Nu testează conștiința sau raționamentul - o mașină poate genera răspunsuri asemănătoare omului, dar nu are o gândire autentică.
- Unii specialiști susțin că acest test stabilește limita de departașare om-mașină prea jos, deoarece chiar și IA simplă poate păcăli oamenii pentru interacțiuni scurte.

În legătură cu programele de IA care au trecut testul Turing, despre unele chatbot-uri IA, cum ar fi Eugene Goostman (2014), realizatorii lor au susținut că programele lor îl trec în condiții specifice, dar nici un software de IA nu prezintă astăzi inteligență generală egală cu a unui om.

Urmând cele menționate anterior, un dispozitiv pentru a acționa ca un om trebuie să aibă următoarele aptitudini: să poată procesa limbajul natural, să raţioneze automat, să înveţe, să poată reprezenta cunoştinţele, la care se pot adăuga aptitudini video şi robotice

***A acţiona raţional***

Acest atribut a dispozitivelor de IA se referă la luarea unor decizii care să ducă la cea mai bună soluţie, sau cea mai bună soluţie posibilă. Acest atribut se referă la:

- A acţiona în funcţie de condiţiile prezente, bazat pe logică și experienţă.
- A nu acţiona sub impulsul momentului, conduşi de sentimente.

Acest atribut a fost folosit în primele aplicații de IA din domeniul energeticii, și anume prin dezvoltarea unui sistem expert (SE) care ajută operatorii din centrale nuclearo-electrice pentru a lua cele mai bune decizii.

În prezent IA este clasificată în trei tipuri:
Narrow AI (IA îngustă) care este caracterizată prin:

- Specific pentru sarcini concrete, limitate – Proiectat pentru o anumită funcție (de exemplu, recunoaștere a imaginii, chatbot, sisteme de recomandare).
- Fără conștientizare de sine – Nu are emoții, conștiință sau înțelegere reală.
- Bazat pe reguli sau bazat pe învățare – Funcționează folosind algoritmi, învățarea automată sau modele de învățare profundă.
- Obișnuit în viața de zi cu zi - Găsit în motoarele de căutare, asistenții virtuali (Siri, Alexa) și vehiculele autonome.

Generalized AI - AGI (IA generalizată) se referă la sisteme de IA, care au caracteristici cognitive specifice ființelor umane, ceea ce înseamnă că pot învăța, raționa și aplica cunoștințele în diferite domenii ca un om. Acest tip de IA ar putea îndeplini orice sarcină intelectuală pe care o poate face un om. În momentul de față AGI nu există încă, dar cercetătorii lucrează la el. Dacă ar fi realizat, ar putea revoluționa domenii precum medicina, inginerie și educație. Trăsături cheie ale AGI:

- Capacitatea de a înțelege și de a învăța din experiențe precum oamenii.
- Abilități generale de rezolvare a problemelor în mai multe domenii.
- Auto-îmbunătățire și adaptare.

Artificial Superinteligence – ASI (Superinteligență artificială) se referă la IA care depășește inteligența umană în toate aspectele, inclusiv creativitatea, rezolvarea problemelor și emoțiile. Astfel de sisteme ar fi capabile de conștientizarea de sine, gândirea independentă și, eventual, chiar conștiință. În prezent, ASI nu există, iar dezvoltarea sa potențială ridică preocupări etice, cum ar fi siguranța AI, alinierea la valorile umane și riscuri existențiale. Specialiștii din domeniu specifică despre ASI următoarele:

- Depășește inteligența umană în toate domeniile.
- Conștientizarea de sine și luarea autonomă a deciziilor.
- Potențial capabil de auto-replicare și inovare dincolo de capacitatea umană.

## 1.2 Istoria Inteligenței Artificiale

Ideile fundamentale ale IA își au originea în Grecia antică, când Aristotel a pus bazele logicii formale, care ulterior va influența raționamentul folosit în algoritmii de IA.

Între secolele XVII-XIX, filosofi precum René Descartes, Leibniz și George Boole au explorat ideea raționamentului mecanizat și a logicii simbolice, care sunt folosite și azi în tehnici de IA. Tot în această perioadă, mai exact în anii 1830 Charles Babbage și Ada Lovelace au conceptualizat prima mașină programabilă (motorul analitic). Acest concept va duce la un dispozitiv real abia în secolul XX.

Nașterea conceptelor de bază ale IA a avut loc între anii 1940-1950, astfel:

- Neuropsihologul american W. McCulloch şi matematicianul american W. Pitts în 1943, folosindu-se de patru surse, și anume informaţii legate de funcţionarea neuronilor, bazele psihologiei, logica predicatelor şi teoria computaţională au dezvoltat primul model al neuronului artificial. Acest neuron avea starea de "on" sau "off". Starea de funcţiune a neuronului era influențată de neuronii învecinaţi.
- În 1949 D. Hebb propune o nouă regulă de modificare a conexiunilor dintre neuroni, ducând la dezvoltarea unui algoritm de învăţare, care ulterior a primit numele de învăţarea Hebbiană.
- În 1950 A. Turing va concepe un test care ulterior îi va purta numele, și este folosit și azi pentru a determina dacă un dispoziţiv acționează ca un om.
- În 1951, Marvin Minsky și Dean Edmonds au construit primul computer bazat pe o rețea neuronală, numit SNARC (Stochastic Neural Analog Reinforcement Calculator). Aceasta a fost o încercare timpurie de a simula o rețea neuronală, folosind 40 de neuroni artificiali. Cu toate acestea, nu a fost o rețea neuronală tradițională, ci mai degrabă un calculator analogic conceput pentru a imita procesele neuronale.
- În 1956 ia naştere termenul de Inteligenţă Artificială, introdus de J. McCharty în cadrul unei conferinţe de 2 luni dintre 10 oameni de ştiinţă din domeniu ai vremii. Întâlnirea nu a rezultat cu descoperiri majore, dar s-au formulat legile și principiile IA, respectiv scopul – realizarea unor maşini care folosesc limbajul, rezolvă probleme caracteristice oamenilor. Atunci a fost înființată IA ca știință.

Perioada aniilor 1950-1970 este considerată prima înflorire a IA, deoarece s-au continuat descoperirile în domeniul calculului neuronal, dar și prin apariția sistemelor expert. Într-adevăr, în 1958 Frank Rosenblatt a dezvoltat Perceptronul, o rețea neuronală timpurie. Mai mult, în 1957 Allen Newell și Herbert A. Simon au dezvoltat General Problem Solver (GPS). Acesta a fost un program IA timpuriu dezvoltat pentru a rezolva o gamă largă de probleme folosind un proces

de raționament pas cu pas. GPS-ul a fost una dintre primele încercări de a crea o mașină universală de rezolvare a problemelor bazată pe raționamentul uman.

La IBM, în S.U.A. au fost conduse cercetări în domeniul IA de către Nathaniel Rochester și echipa sa, dezvoltând unele dintre primele programe IA pentru computere. Echipa sa a lucrat la învățarea automată, raționamentul bazat pe logică și rețele neuronale. Rochester a ajutat, de asemenea, la organizarea Conferinței Dartmouth din 1956. Ulterior, tot la IBM, Arthur Samuel a dezvoltat unul dintre primele programe de auto-învățare: un IA care juca jocul dame. El a folosit învățarea prin întărire pentru a-și îmbunătăți strategia în timp, un concept care a inspirat cercetările ulterioare în învățarea automată.

Samuel a declarat atunci:

„Învățarea automată este domeniul de studiu care oferă computerelor capacitatea de a învăța fără a fi programate în mod explicit.”

O concepție greșită apărută în această perioadă este că

„Programele pot face doar ceea ce poate creatorul lor”.

Această frază nu reprezintă cu exactitate opera lui Samuel. De fapt, programul său de dame îl putea depăși, demonstrând că IA ar putea învăța dincolo de cunoștințele creatorului său. Ideea că „IA poate face doar ceea ce este programat să facă” este parțial adevărată, deoarece învățarea automată permite programului să se adapteze și să se îmbunătățească dincolo de programarea sa inițială. Munca lui Samuel A. a pus bazele pentru IA modernă, învățarea profundă și învățarea prin consolidare / întărire.

În 1958, la MIT, J. McCharty a dezvoltat primul limbaj de IA, denumit LISP (LISt Programming), care a dominat toate limbajele din domeniu, timp de 30 de ani.

Mai târziu, în anii 1960-1970, oamenii de știință s-au concentrat pe dezvoltarea sistemelor expert bazate pe reguli. Acestea au fost folosite cu precădere în diagnosticul medical și luarea deciziilor. Tot în această perioadă a fost dezvoltat primul chatbot, ELIZA, de către Joseph Weizenbaum.

Perioada următoare, cea între anii 1970-1980 este considerată de către unii specialiști prima iarnă a IA, deoarece finanțarea a scăzut din cauza progresului lent în IA. Mai exact, tot mai multe sisteme de IA eșuau, iar perceptronii care se bazau pe reguli timpurii, nu puteau rezolva probleme complexe. Acest fapt, a dus la reduceri de finanțare din cauza dezamăgirii progresului. Mai mult, raportul

Lighthill din 1973, în care a fost criticată cercetarea IA, a dus la scăderea sprijinului în Marea Britanie.

În 1980 a avut loc o schimbare majoră, și anume s-a trecut de la programe de laborator, create în mediul academic, la programe comerciale. Astfel, primul SE comercial, R1 a fost funcţional și utilizat în 1982 de Digital Equipment Corporation. Programul a ajutat configurarea comenzilor pentru sisteme noi de PC. Până în 1984 programul a ajutat compania să aibă un beneficiu de 40 mil. USD. Până în 1988 cele mai mari companii aveau propriul grup de lucru pe IA. Industria IA a explodat, de la o cifra de afaceri de milioane USD, la miliarde la sfârşitul anilor 80.

În 1986 a fost inventat algoritmul de "back-tracking propagation" (propagare înapoi), care a revoluţionat RNA și algoritmii de învăţare automată

Însă, spre sfârșitul anilor 1980, aceste programe comerciale bazate pe SE au devenit prea scumpe și inflexibile, ceea ce a dus la a doua iarnă IA. Mai mult, puterea de calcul era limitată, iar RNA nu aveau rezultatele dorite. În acești ani nu a fost introdus nimic nou în domeniul IA, ci doar au fost folosite cunoștinţele anterioare.

La început, IA a fost fondată pentru a se opune limitărilor introduse de domeniile existente (controlul și statisticile), dar ulterior, spre sfârșitul anilor 1990, aceste două aspecte au dus la creșterea interesului în IA, și ieșirea din impasul creat anterior. În această perioadă au fost combinate ideile vechi de control și statistică, cu cele noi din IA, ceea ce a rezultat fiind, noi RNA și algoritmi de învățare bazați pe statistică, care sunt folosiți și în prezent cu succes.

În 1995 s-a introdus pentru prima dată după o perioadă lungă de timp un nou termen în IA, și anume acela de agent virtual, care este definit ca un program capabil să rezolve o problemă găsind cea mai bună soluţie.

Perioada care începe din anii 2000 și până în prezent, este considerată revoluția modernă a IA.

Din 2001 atenţia s-a îndreptat asupra stocării datelor şi manipularea acestora (storage and data mining). Mai mult, după 2010 a apărut o explozie de aplicații și algoritmi în domeniul învățării profunde, respectiv în manevrarea bazelor de date de dimensiuni mari.

În 2007, fondatorii IA au făcut o caracterizare a noilor tendinţe IA, și au ajuns la concluzia că s-a deviat de la ideea inițială, ajungându-se la rezolvarea unor probleme foarte specializate, "uitând de scopul iniţial al IA".

Anii 2020 sunt caracterizați de programe de IA de tipul chatbot-urile alimentate de IA (GPT, Bard), vehiculele autonome și sistemele IA generative.

Aceste tendințe transformă industriile din societățile umane și în unele situații produc neliniște.

Fig. 1.1. ilustrează sub forma unei diagrame, principalele repere în istoria IA.

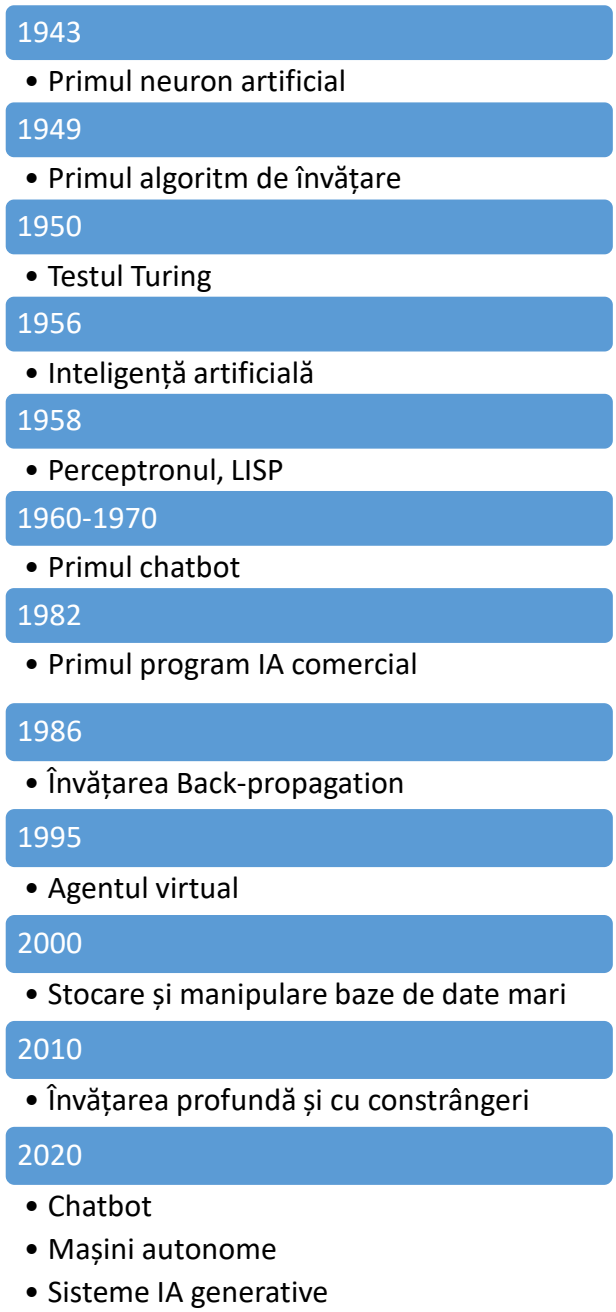



**Fig. 1.1**. Diagrama istoriei IA

### 1.3 Tehnicile de IA

În prezent sunt un număr mare de tehnici de IA, care pot fi împărțite în nouă clase. Următoarele paragrafe prezintă succint aceste clase de IA.

Învățarea automată (învățarea mașinii) – Machine Learning (ML)

Această categorie se referă la algoritmi de IA, care dau posibilitatea computerelor să învețe folosind date și să facă predicții respectiv, să ia decizii. O listă completă a acestor algoritmi cuprinde: învățare supravegheată, regresia liniară, regresia logistică, suport vector machines (SVM), arbori de decizie și pădure aleatorie, k cei mai apropiați vecini (k-NN), rețele neuronale, învățare nesupravegheată, k-means clustering, clustering ierarhic, analiza componentelor principale (PCA), autoencodere, învățare semi-supravegheată, învățare prin consolidare (RL), q-learning, deep q-networks (DQN), metode de gradient de politică, metode actor-critice.

Învățarea profundă – Deep Learning (DL)

DL este o subcategorie a ML, dar este de multe ori privită diferit, deoarece cuprinde doar rețele neuronale artificiale, iar algoritmii de învățare sunt dedicați acestora. Cele mai uzuale aplicații sunt în recunoașterea imaginii, a formelor, respectiv sisteme autonome. În această categorie intră următoarele: rețele neuronale convoluționale (CNN), rețele neuronale recurente (RNN), memoria pe termen lung (LSTM), transformatoare (de exemplu, BERT, GPT), rețele adversare generative (GAN) și învățare auto-supravegheată.

Algoritmi evolutivi și bio-inspirați

Acești algoritmi sunt inspirați din procese biologice, fiind aplicați în probleme de optimizare. Printre cele mai populare tehnici IA din această categorie sunt algoritmii genetici (GA), optimizarea roiului de particule (PSO), optimizarea coloniilor de furnici (ACO), evoluția diferențială (DE), colonia de albine artificiale (ABC).

Inteligența artificială probabilistică și statistică

Această categorie de IA utilizează teoria probabilității și a statisticii pentru a rezolva problema incertitudinii din sistemele de IA. În această categorie intră

rețelele Bayesiene, modelele Markov ascunse (HMM), procesele de decizie Markov (MDP) și metodele Monte Carlo.

Sisteme fuzzy și IA hibridă

În această categorie intră sisteme de IA bazate pe logica fuzzy, și sisteme care cuprind mai multe tehnici de IA, de exemplu sisteme neuro-fuzzy.

Sisteme bazate pe cunoștințe și IA simbolică

Această categorie cuprinde sistemele care se bazează pe reguli predefinite și folosirea simbolurilor pentru a reprezenta cunoștințele. Cele mai populare tehnici din această categorie sunt sistemele expert.

Procesarea limbajului natural – Natural Language Processing (NLP)

Încorporarea de cuvinte (Word2Vec, GloVe), atenție mecanisme și transformatoare, tecunoașterea entității denumite (NER) și analiza sentimentelor sunt tehnicile de IA care intră în această categorie.

Computer vision

În această categorie de tehnici de IA intră detectarea obiectelor (YOLO, Faster R-CNN), segmentarea imaginii (U-Net, Mask R-CNN) și recunoașterea feței. Așa cum se poate observa, această categorie se referă la procesarea și înțelegearea imaginilor și a video-urilor.

Planificarea IA și algoritmi de căutare

Această categorie de tehnici de IA se referă la sisteme IA construite special pentru probleme de planificare și organizare, precum și în luarea unor decizii efective. Trei dintre aceste tehnici sunt algoritmul A*, algoritmul Minimax (utilizat în jocuri) și planificarea genetică.

Tabelul 1.1. prezintă în formă tabelară categoriile de tehnici de IA, și exemplele enumerate în paragrafele anterioare.

**Tabelul 1.1** Tehnici de IA

| **Categoria** | **Tehnica** |
|---|---|
| ML | Învățare supravegheată, regresia liniară, regresia logistică, suport vector machines (SVM), arbori de decizie și pădure aleatorie, k-cei mai apropiați vecini (k- |

| | |
|---|---|
| | NN), rețele neuronale, învățare nesupravegheată, k-means clustering, clustering ierarhic, analiza componentelor principale (PCA), autoencodere, învățare semi-supravegheată, învățare prin consolidare (RL), q-learning, deep q-networks (DQN), metode de gradient de politică, metode actor-critice |
| DL | Rețele neuronale convoluționale (CNN), rețele neuronale recurente (RNN), memoria pe termen lung (LSTM), transformatoare (de exemplu, BERT, GPT), rețele adversare generative (GAN) și învățare auto-supravegheată |
| IA probabilistică și statistică | Rețele Bayesiene, modele Markov ascunse (HMM), procesele de decizie Markov (MDP) și metode Monte Carlo |
| Algoritmi evolutivi | Algoritmi genetici (GA), optimizarea roiului de particule (PSO), optimizarea coloniilor de furnici (ACO), evoluție diferențială (DE), colonie de albine artificiale (ABC) |
| Sisteme fuzzy | Sisteme fuzzy, sisteme neuro-fuzzy |
| Sisteme bazate pe cunoștințe | Sistemele expert |
| NLP | Încorporare de cuvinte (Word2Vec, GloVe), atenție mecanisme și transformatoare, tecunoașterea entității denumite (NER) și analiza sentimentelor |
| Computer vision | Detectarea obiectelor (YOLO, Faster R-CNN), segmentarea imaginii (U-Net, Mask R-CNN) și recunoașterea feței |
| Planificarea IA | Algoritmul A*, algoritmul Minimax (utilizat în jocuri) și planificarea genetică |

Dintre aceste tehnici de IA, în cadrul acestui curs se vor prezenta doar trei, și anume sistemele fuzzy, rețelele neuronale artificiale și algoritmii genetici. În cele ce urmează se prezintă sumar caracteristicile acestor tehnici.

Sistemele fuzzy folosesc logica fuzzy pentru a rezolva probleme într-un mod specific oamenilor. Acest lucru este posibil deoarece, LF copiază modul rațional de gândire al oamenilor prin folosirea simbolurilor și a raționamentului specific uman. Comparativ cu logica clasică, care are doar două valori posibile – Adevărat

şi Fals, logica fuzzy lucrează cu o infinitate de valori între extremele Adevărat – 1, și Fals – 0. Prin această abordare se simulează modul în care apar informaţiile reale în procesele reale. Prima aplicaţie notabilă a sistemelor fuzzy în managementul energiei este cea realizată în Japonia pentru a îmbunătăţi confortul și precizia curselor în trenurile de mare viteză.

Rețele neuronale artificiale simulează felul în care funcţionează creierul uman. Astfel, RNA conțin neuroni, care au caracteristicile neuronilor din componența sistemului nervos uman. Principalul punct forte al RNA este capacitatea de a învăța. Dintre cele trei tehnici de IA, doar RNA sunt capabile să realizeze această activitate specifică oamenilor. În ceea ce privește dezvoltarea RNA, există varianta soft (programe informatice, fără un suport fizic dedicat) și varianta hard (microprocesoare dedicate doar pentru simularea RNA). În prezent, cele mai populare variante de construcție sunt cele sub formă de programe informatice, deoarece utilizarea unor microprocesoare folosite doar pentru a implementa o RNA sau neuroni individuali este foarte scumpă. Există multe aplicații ale RNA folosite cu succes, în special în problemele de recunoaştere (imagini), control (vehicule) etc.

Algoritmii genetici funcționează pe baza unui algoritm care copiază fenomenul de evoluţie genetică. Așa cum s-a văzut mai sus în capitol, aceste tehnici fac parte din categoria metodelor de căutare evolutive. În consecință, ele folosesc terminologia din genetică și sunt inspirate de procesele evolutive care au loc în natură. Aceste tehnici de IA sunt folosite pentru a rezolva probleme de căutare a unei soluţiei optime dintr-un bagaj mare de soluții. Primele aplicații ale AG a apărut la sfârşitul anilor 1980, când General Electric a comercializat primul program bazat pe AG, dedicat proiectării proceselor industriale.

## 1.4 Domeniile de aplicație ale IA

Domeniile de aplicație ale IA sunt foarte variate, iar în unele situații aplicațiile au ajuns să depășească inteligența umană. Cele mai recunoscute domenii de aplicație sunt recunoașterea vocii, participarea la jocuri, lupta împotriva spam-urilor, robotica, traducere, agenţi virtuali, recunoaşterea activităţii, biometrie, realitate virtuală, creativitate artificială și înţelegearea limbajului natural.

Un exemplu timpuriu este în programarea logisticii – în 1991 armata S.U.A. a folosit o aplicaţie IA pentru realizarea programării logisticii şi transportului de oameni şi materiale.

În ceea ce privește participarea la jocuri, în 1997 Deep Blue de la IBM l-a învins pe campionul de șah Garry Kasparov, dovedind abilitatea IA în gândirea strategică. De asemenea, în 2016 AlphaGo l-a învins pe campionul mondial Go, demonstrând abilitățile strategice avansate ale IA.

În 2011, IBM Watson a câștigat Jeopardy!, prezentând procesarea limbajului natural al IA.

În prezent, există autoturisme autonome comerciale, dar în ceea ce privește subiectul autoturismelor robotizate, în 2005 o mașină robotizată a câștigat cursa DARPA Grand Challenge.

## 1.5 Utilizarea IA în managementul energiei

Pentru a înțelege utilizarea aplicațiilor de IA în managementul energiei prima etapă este definirea acestui concept. Astfel, managementul presupune activitatea sau arta de a conduce procesele unei companii. Mai mult, este caracterizată ca știința organizării și conducerii întreprinderii.

În domeniul energeticii, managementul se referă la ansamblul activităţilor de organizare, conducere şi de gestiune a proceselor şi companiei din domeniul energiei (energiei electrice). Pentru aceasta există un sistem de management al energiei conform ISO 50001, unde sunt specificate următoarele, în legătură activitățile unei companii în demersul managementului energiei:

- Urmăreşte optimizarea consumurilor de resurse şi identificare de surse alternative de energie.
- Promovează reducerea costurilor prin creşterea eficienţei şi performanţelor în utilizarea energiei.
- Demonstrează angajamentul de reducere a emisiilor de dioxid de carbon către comunitate și autorităţi.
- Implică angajaţii în identificarea de oportunităţi pentru eficientizarea consumului de energie și identificarea de soluţii alternative.

În prezent aplicații ale IA sunt implicate în toate ramurile energeticii:

- Producerea energiei electrice – prognoza producerii de energie electrică, controlul centralelor electrice.
- Transportul energiei electrice – decongestionarea funcționării sistemului de transport al energiei electrice.
- Distribuţia energiei electrice – controlul calității energiei electrice, identificarea și clasificarea problemelor de calitate a energiei electrice.

- Utilizarea energiei electrice – controlul proceselor de producție în companii.
- Susţinerea proceselor energetice – mentenanța proceselor energetice, diagnosticarea problemelor apărute în procesele energetice.

## 1.6 Intrebări și exerciții

1. Care sunt tehnicile de IA care lucrează cu reprezentări simbolice?
   Sisteme expert / rețele neuronale artificiale / sisteme fuzzy

2. În câte categorii se împart tehnicile de IA?
   3 / 7 / 9

3. În prezent s-au descoperit sisteme de IA generalizată.
   Adevărat / Fals

4. Dintre LF, RNA și AG, RNA sunt primele tehnici de IA.
   Adevărat / Fals

# 2

# Logica fuzzy. Mulțimi fuzzy

Acest capitol face o introducere în logica fuzzy, respectiv sistemele și controlul fuzzy. Astfel, capitolul începe cu istoria logicii fuzzy, apoi prezintă elementele de bază ale acesteia și anume mulțimile fuzzy. Capitolul se încheie cu o listă de exerciții și întrebări pentru autoevaluare.

## 2.1 Istoria logicii fuzzy

Istoria logicii fuzzy începe cu Aristotel (384–322 î.Hr.), considerat părintele logicii, care a pus bazele logicii, ca știință. Atunci el a introdus logica binară, Adevărat (1) și Fals (0). Adică, un element, o afirmație poate fi doar adevărată sau falsă, nu există nimic între aceste două valori. Ulterior, abia în secolul XX a apărut ideea de logică multi-valentă, adică existența unor valori intermediare, sau diferite de 0 și 1.

În 1917 Jan Łukasiewicz a introdus logica cu trei valori (logica trivalentă), care este caracterizată de Adevărat, Fals și o valoare intermediară precum Nedeterminat sau Posibil. Patru ani mai târziu, în 1921, Emil Post a extins logica trivalentă la logica cu ”n” valori, mai exact, un număr arbitrar de valori între adevărat și fals.

Un alt om de știință care a lucrat cu logica multi-valentă a fost Donald Knuth. Acesta între anii 1970-1980 a studiat logica tetravalentă (patru valori de adevăr), care a fost explorată mai târziu în „Arta programarii computerelor”. El a folosit logica tetravalentă pentru aplicații în sistemele de calcul, în special atunci când se ocupă de incertitudine și contradicții în programare și circuite computerizate.

Părintele logicii fuzzy este considerat Lotfi Zadeh, care a introdus pentru prima dată logica fuzzy, el formalizând logica cu mai multe valori într-un interval continuu (de la 0 la 1), permițând grade de adevăr în sisteme.

O sinteză a evoluției logicii fuzzy se poate observa în fig. 2.1.

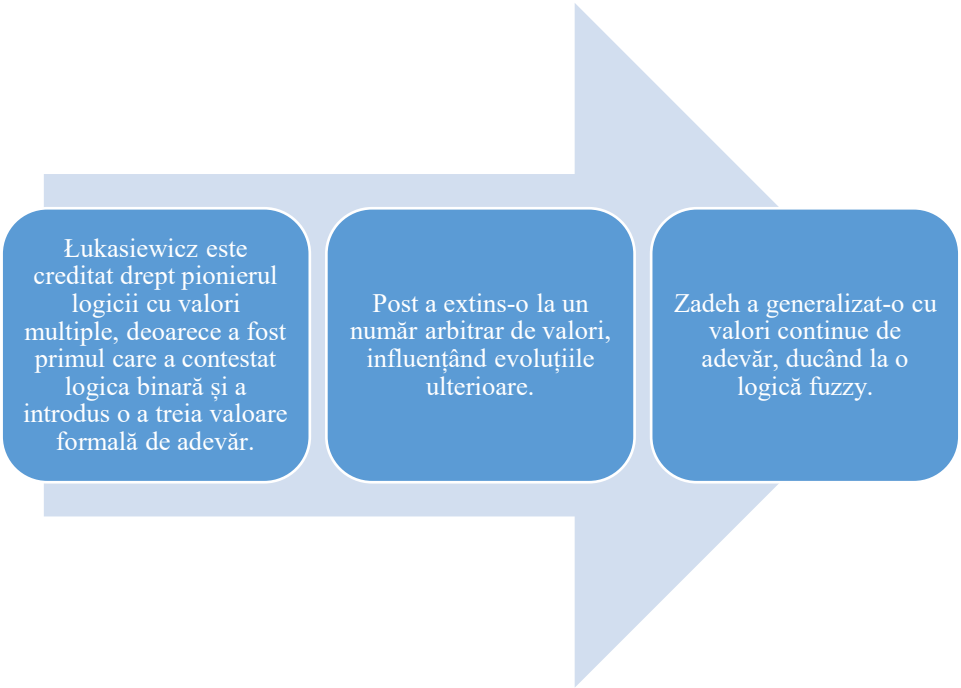



**Fig. 2.1**. Evoluția logicii fuzzy

Logica fuzzy (LF) a fost introdusă în 1965 de către Lotfi A. Zadeh, profesor la Universitatea din California, Berkeley. El a dezvoltat conceptul ca o extensie a logicii booleene clasice pentru a gestiona incertitudinea și imprecizia, care sunt comune în problemele din lumea reală. O istorie cronologică mai detaliată a logicii fuzzy este prezentată în paragrafele următoare:

- 1965 – Introducerea mulțimile fuzzy

Zadeh a publicat lucrarea revoluționară, „Fuzzy Sets", în jurnalul Information and Control. El a propus că, spre deosebire de logica binară tradițională (adevărat/fals), elementele dintr-o mulțime ar putea avea grade de apartenență cuprinse între 0 și 1. Aceasta a introdus o modalitate de a reprezenta date vagi în mod matematic.

- Anii 1970 – Progrese teoretice

Cercetătorii au extins activitatea lui Zadeh, perfecționând teoria mulțimilor fuzzy și dezvoltând relațiile fuzzy, aritmetica fuzzy și sistemele de control fuzzy. În acești ani au început să apară aplicații în recunoașterea modelelor și luarea deciziilor.

- Anii 1970 – Anii 1980 – Aplicații industriale în Japonia

Japonia a devenit lider în aplicarea logicii fuzzy în lumea reală, în special în sisteme de control. Primul succes major a fost un sistem de control la metroul din

Sendai (1987), care a îmbunătățit semnificativ confortul călătoriei. Companii precum Mitsubishi, Hitachi și Omron au aplicat LF în produsele de consum, cum ar fi aparatele de aer condiționat, mașinile de spălat și camerele foto.

- Anii 1990 – Extinderea globală și popularitatea

Logica fuzzy a câștigat o adoptare pe scară largă în sistemele de control, robotică și inteligență artificială. Astfel, controlul fuzzy a fost folosit în sisteme de frânare antiblocare (ABS), transmisii automate, ascensoare și aparate de uz casnic. IEEE a introdus chiar și conferințe și reviste dedicate cercetării cu logica fuzzy.

- Anii 2000 – Integrare cu IA și Machine Learning

Odată cu creșterea domeniului de inteligență artificială, logica fuzzy a fost integrată cu rețele neuronale artificiale, algoritmi genetici și sisteme expert. Această fuziune a îmbunătățit luarea deciziilor, optimizarea și gestionarea incertitudinii în sisteme complexe.

Avantajele logicii fuzzy sunt:

1. Se ocupă de incertitudine - Spre deosebire de logica clasică, logica fuzzy imită raționamentul uman.
2. Îmbunătățește procesul decizional – Aplicat în AI, sisteme de control și automatizare.
3. Îmbunătățește optimizarea – Folosit în sisteme de alimentare, robotică și finanțe.

## 2.2 Teoria mulțimilor fuzzy

Despre mulțimile fuzzy L. Zadeh a spus:

> "Noţiunea de mulţime fuzzy constituie un punct de plecare convenabil în dezvoltarea unui cadru conceptual asemănător celui utilizat în teoria mulţimilor clasice dar care este mult mai general decât acesta şi care, potenţial, poate avea domenii de aplicabilitate mult mai largi. Un astfel de cadru furnizează un mod natural de abordare a problemelor în care sursa impreciziei este mai degrabă absenţa unor criterii exacte privind apartenenţa la o anumită clasă decât prezenţa unor variabile aleatorii."

Pentru a înțelege mai bine teoria mulțimiilor fuzzy, în continuare se va face o descriere succintă a teoriei mulțimilor clasice și a operațiilor asociate acestora.

În teoria mulțimilor clasice, o mulțime este o colecție bine definită de elemente. În consecință, o mulțime clasică conține un număr de elemente

distincte, care pot fi numere, litere, obiecte etc., iar calitatea de membru a mulțimii este clară, deci, un element fie aparține, fie nu aparține mulțimii. Relația de apartenență a unui element are valoarea 1 sau 0.

***Exemplu***

Fie mulțimea A = {linie, transformator, generator}.

Se consideră elementele ”linie” și ”turbină”

În raport cu mulțimea A, elementul ”linie” are relația de apartenență 1, dar elementul ”turbină” are relația de apartenență 0.

În teoria mulțimilor clasice sunt operații care combină sau manipulează mulțimile. Printre acestea, cele mai folosite sunt apartenența, egalitatea, reuniunea, intersecția, complementaritatea, disjuncția, cardinal, submulțime și diferența.

- Reuniunea – mulțimea elementelor care se află fie în una dintre mulțimi, fie în ambele.
- Intersecția – mulțimea elementelor care se află în ambele mulțimi.
- Diferența – mulțimea elementelor care se află în una dintre mulțimi, dar nu și în cealaltă.
- Complement – mulțimea elementelor care nu se află în mulțimea analizată, comparativ cu o mulțime etalon (universală).
- Submulțime – mulțimea elementelor care conține un număr de elemente dintr-o mulțime etalon.
- Apartenența – un element aparține unei mulțimi dacă relația de apartenență față de aceea mulțime este 1. O mulțime aparține altei mulțimi, și se numește submulțimea ei, dacă toate elementele submulțimii sunt în mulțimea etalon.
- Egalitate – două mulțimi sunt egale, dacă cele două mulțimi au aceleași elemente.
- Disjuncție – două mulțimi sunt disjuncte, dacă diferența lor este mulțimea nulă (o mulțime care nu conține nici un element).
- Cardinal – cardinalul unei mulțimi este numărul de elemente ale mulțimii.

În teoria mulțimilor fuzzy, o mulțime este o colecție de elemente care sunt caracterizate prin faptul că apartenența la mulțime este definită printr-o funcție de apartenență care ia valori în intervalul închis [0,1]. Acest aspect este prezentat

grafic și matematic prin fig. 2.2 și relația (2.1). Se poate observa că, mulțimea are 9 elemente, funcția de apartenență a mulțimii are forma unui triunghi neregulat, iar funcția de apartenență a elementului de pe poziția 6 are valoarea 0,8.

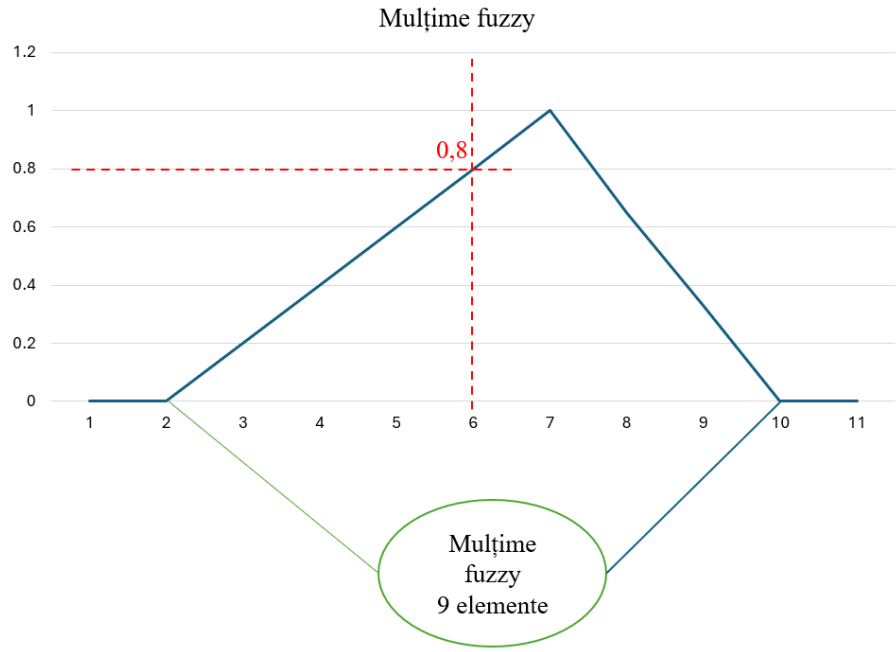



**Fig. 2.2**. Apartenența la o mulțime fuzzy

$$\forall x \in \boldsymbol{E}: \mu_{\boldsymbol{E}}(x) \in [0,1] \tag{2.1}$$

Din paragrafele anterioare reiese că principala diferență dintre o mulțime clasică și o mulțime fuzzy este faptul că elementele mulțimii fuzzy au atașată o valoare a funcției de apartenență care ia valori între 0 și 1, contrar relației de apartenență a elementelor unei mulțimi clasice, care are valoarea doar 0 sau 1. De altfel, dacă se consideră o mulțime clasică, **E**, atunci mulțimea fuzzy **A** atașată acesteia este descrisă sub forma unei perechi

$$\mathbf{A} = \{(x, \mu_{\mathbf{A}}(x)), x \in \mathbf{E}\}, \quad \mu_{\mathbf{A}} \in [0,1] \tag{2.2}$$

***Exemplu***

Având mulțimea clasică E = {a, b, c, d, e, f}

O mulțime fuzzy atașată acesteia este ilustrată în tabelul 2.1

**Tabelul 2.1.** Mulțimea fuzzy **A** atașată mulțimii clasice **E**

| | a | b | C | d | e | f |
|---|---|---|---|---|---|---|
| A= | 0.615 | 0.09 | 0 | 1 | 0.5 | 0.87 |

Trebuie specificat despre funcțiile de apartenență ale elementelor unei mulțimi fuzzy, că ele indică o incertitudine nonprobabilistică. Mai exact, la o mulțime fuzzy, elementele ei aparțin clar acesteia, dar gradul de incertitudine a lor este variabil. Această caracteristică a LF, fac din ea o tehnică de IA care oferă posibilitatea de a lucra cu date incerte, incomplete, care sunt caracteristice proceselor energetice reale.

Un aspect important în teoria mulțimilor fuzzy este reprezentarea lor, care se poate face sub formă grafică, analitică, folosind perechi, fracție simbolică sau integrală simbolică.

Prin folosirea reprezentării grafice, elementele și valorile funcțiilor de apartenență se reprezintă grafic, așa cum este exemplificat în fig. 2.3, pentru o mulțime fuzzy **A**, care conține elementele din intervalul [a1, a8], iar acestea aparțin axei 0x (mulțimii **X**), iar $\mu_A(x)$ este funcția de apartenență.

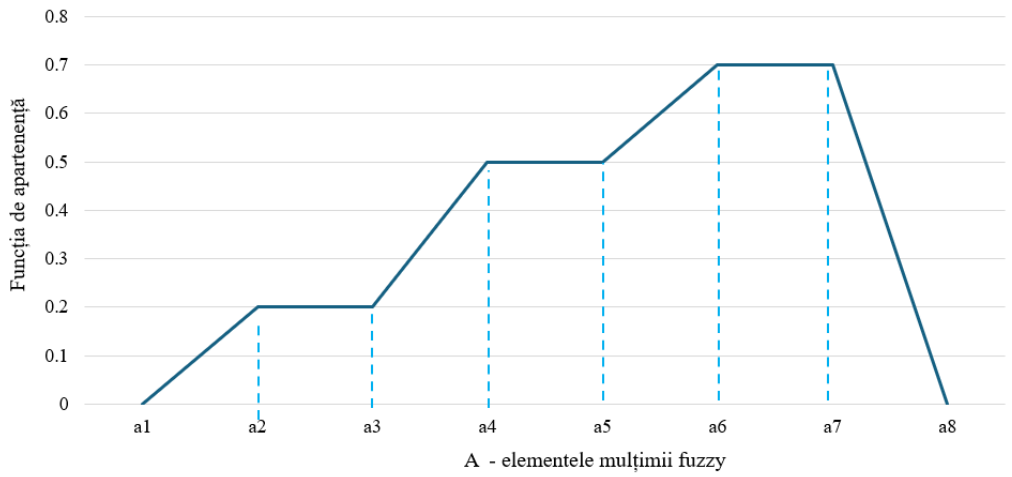



**Fig. 2.3**. Reprezentarea grafică a unei mulțimi fuzzy complexe

În cazul reprezentării analitice, funcția de apartenență și elementele mulțimii fuzzy sunt definite print-o formă analitică, care particularizată pentru exemplu din fig. 2.3. are forma din relația (2.3), **A**={(x, $\mu_A$(x))| x∈**X**}.

$$\mu_A(x) = \begin{cases} 0, x < a_1 \\ \dfrac{x - a_1}{a_2 - a_1} \cdot 0{,}2, a_1 \leq x < a_2 \\ 0{,}2, a_2 \leq x < a_3 \\ \dfrac{x - a_3}{a_4 - a_3} 0{,}5, a_3 \leq x < a_4 \\ 0{,}5, a_4 \leq x < a_5 \\ \dfrac{x - a_5}{a_6 - a_5} 0{,}7, a_5 \leq x < a_6 \\ 0{,}7, a_6 \leq x < a_7 \\ \dfrac{a_8 - x}{a_8 - a_7} 0{,}7, a_7 \leq x < a_8 \end{cases} \tag{2.3}$$

Reprezentarea prin intermediul unor perechi presupune definirea unei mulţimi de perechi ordonate, fiecare pereche fiind formată dintr-un element al mulţimii fuzzy şi gradul de apartenenţă al acestuia la mulţime **A**={(x, $\mu_A$(x))| x∈**X**}, care în cazul exemplului folosit anterior are forma din relația (2.4).

**A**={($a_1$, 0), ($a_2$, 0,2), ($a_3$, 0,2), ($a_4$, 0,5), ($a_5$, 0,5), ($a_6$, 0,7), ($a_7$, 0,7), ($a_8$, 0)} (2.4)

Dacă domeniul de definiție este discret și finit, reprezentarea mulțimii fuzzy se poate face printr-o sumă simbolică, fiecare element al sumei reprezentând o fracţie care are ca numitor un element al mulţimii fuzzy, iar ca numărător gradul de apartenenţă al elementului, așa cum este ilustrat în relația (2.5), iar exemplu numeric prin relația (2.6).

$$A = \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \ldots = \sum_{i=1}^{n} \frac{\mu_A(x_i)}{x_i} \tag{2.5}$$

$$A = \frac{0}{a_1} + \frac{0{,}1}{\frac{a_1 + a_2}{2}} + \frac{0{,}2}{a_2} + \frac{0{,}2}{a_3} + \frac{0{,}325}{\frac{a_3 + a_4}{2}} + \frac{0{,}5}{a_4} + \frac{0{,}5}{a_5} + \frac{0{,}625}{\frac{a_7 + a_6}{2}} + \frac{0{,}7}{a_6} + \frac{0{,}7}{a_7} + \frac{0}{a_8} \tag{2.6}$$

În cazul în care domeniul de definire a elementelor mulțimii fuzzy este continuu și infinit, o metodă de reprezentare a mulțimii este prin intermediul unei integrale simbolice, precum arată relația (2.7), iar exemplu numeric în relația (2.8).

$$A = \left\{ \int \frac{\mu_A(x)}{x} \right\} \tag{2.7}$$

$$A = \int_{a_1}^{a_6} \frac{\mu_A(x)}{x} \tag{2.8}$$

În teoria mulțimilor fuzzy, funcțiile de apartenență ale elementelor mulțimiilor sunt caracterizate prin intermediul unor termeni specifici, care sunt enumerați și explicați în continuare:

- ***Suportul*** – totalitatea elementelor care au funcţia de apartenenţă diferită de zero.
- ***Lăţimea*** (lărgimea) – distanța între expremitățile mulțimii fuzzy. Exemplu - Width(A)=a6-a1
- ***Nucleul*** – mulţimea elementelor pentru care funcţia de apartenenţă are valoarea 1.
- ***Înălţimea*** – valoarea maximă a funcţiei de apartenenţă.
- ***Vecinătatea*** – reprezintă submulţimea care conţine elementele care nu sunt în nucleu. Vecinătatea de stânga și vecinătatea de dreapta.
- ***Tăietura*** – unei mulţimi fuzzy **A** la un nivel *t*, reprezintă mulţimea elementelor care au gradele de apartenenţă cel puţin egale cu tăietura.
- ***Cardinalul*** (scalarul) – suma gradelor de aparteneţă a tuturor elementelor.

### 2.3 Operații cu mulțimi fuzzy

Proprietățile algebrice ale mulțimilor fuzzy sunt aceleași ca a celor din teoria mulțimilor clasice: reuniunea, intersecția, diferența etc. Diferența constă în faptul că, în cazul mulțimilor fuzzy impactul funcțiilor de apartenență asupra rezultatului final este mare. În cele ce urmează, se vor prezenta operațiile caracteristice mulțimilor clasice, modificate pentru a corespunde mulțimilor fuzzy, precum și operații particulare mulțimilor fuzzy, deci care nu se întâlnesc în teoria mulțimilor clasice.

***Egalitate***

Două mulțimi fuzzy sunt egale dacă au aceleași elemente, iar funcțiile de apartenență ale acestora sunt la fel, iar valorile lor identice, adică

$$\mu_A(x) = \mu_B(x). \tag{2.9}$$

***Submulțime***

O mulțime fuzzy este considerată ca fiind o submulțime a altei mulțimi fuzzy (denumită mulțime mare), dacă prima mulțime este inclusă în a doua mulțime (toate elementele submulțimii sunt în mulțimea mare), iar valorile funcției de apartenență a elementelor submulțimii sunt mai mici decât valorile mulțimii mari, adică

$$\mu_A(x) \leq \mu_B(x). \tag{2.10}$$

***Intersecție***

Intersecția a două mulțimi fuzzy este o o nouă mulțime fuzzy, care conține elementele celor două mulțimi, iar valorile funcției de apartenență a noii mulțimi este valoarea minimă dintre cele două valori (ale funcțiilor de apartenență a celor două mulțimi), adică

$$\mu_C(x) = \min(\mu_A(x), \mu_B(x)) = \mu_A(x) \wedge \mu_B(x), \tag{2.11}$$

unde C este noua mulțime fuzzy.

***Reuniune***

Reuniunea a două mulțimi fuzzy este o o nouă mulțime fuzzy, care conține elementele celor două mulțimi, iar valorile funcției de apartenență a noii mulțimi este valoarea maximă dintre cele două valori (ale funcțiilor de apartenență a celor două mulțimi), adică

$$\mu_C(x) = \max(\mu_A(x), \mu_B(x)) = \mu_A(x) \vee \mu_B(x), \tag{2.12}$$

unde C este noua mulțime fuzzy.

***Complementaritate***

Complementara unei mulțimi fuzzy este o nouă mulțime fuzzy, care are aceleași elemente, dar valorile funcției de apartenență sunt obținute prin diferența dintre 1 și acestea, mai exact

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x) \tag{2.13}$$

***Sumă algebrică***

Suma algebrică a două mulțimi fuzzy este o nouă mulțime fuzzy, care conține elementele celor două mulțimi, iar valorile funcțiilor de apartenență a elementelor noii mulțimi se obțin prin efectuarea operației

$$\mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x) \tag{2.14}$$

***Produs algebric***

Produsul algebric a două mulțimi fuzzy este o nouă mulțime fuzzy, care conține elementele celor două mulțimi, iar valorile funcțiilor de apartenență a elementelor noii mulțimi se obțin prin efectuarea operației

$$\mu_{A \cdot B}(x) = \mu_A(x) \cdot \mu_B(x) \tag{2.15}$$

***Sumă mărginită (specifică mulțimilor fuzzy)***

Suma mărginită a două mulțimi fuzzy este o nouă mulțime fuzzy, care conține elementele celor două mulțimi, iar valorile funcțiilor de apartenență a elementelor noii mulțimi se obțin prin efectuarea operației

$$\mu_{A \oplus B}(x) = min\{\, 1, \mu_A(x) + \mu_B(x)\} \tag{2.16}$$

***Diferență mărginită (specifică mulțimilor fuzzy)***

Diferența mărginită a două mulțimi fuzzy este o nouă mulțime fuzzy, care conține elementele celor două mulțimi, iar valorile funcțiilor de apartenență a elementelor noii mulțimi se obțin prin efectuarea operației

$$\mu_{A(-)B}(x) = max\{\, 0, \mu_A(x) - \mu_B(x)\} \tag{2.17}$$

Operatori specifici teoriei mulțimilor fuzzy sunt operatorii de întărire sau modificare, care sunt folosiți în cazul în care se lucrează cu termeni lingvistici.

***Normalizare***

Operația de normalizare constă în împărțirea valoarea funcției de apartenență la valoarea maximă a funcției de apartenență, adică înălțimea mulțimii fuzzy. Relația matematică este

$$Norm(A) = \frac{\mu_A}{h(A)} \tag{2.18}$$

***Concentrare***

Operația de concentrare a unei mulțimi fuzzy se referă la valoarea funcției de apartenență ridicată la pătrat. Relația următoare ilustrează acest operator.

$$Con(A) = A^2 = \mu_A^2(x) \tag{2.19}$$

***Dilatare***

Operația de dilatare a unei mulțimi fuzzy constă în realizarea operației de radical a valorilor funcției de apartenență. Relația (2.20) descrie operația de dilatare.

$$Dil(A) = \sqrt{A} = \sqrt{\mu_A(x)} \tag{2.20}$$

În literatura de specialitate au fost propuse metode prin care se definesc noi mulțimi fuzzy prin folosirea operatorilor definiți anterior. Aceste mulțimi sunt enumerate prin relațiile (2.21) – (2.25).

Foarte A $$Con(A) = A^2 = \{x, \mu_A^2(x)\} \tag{2.21}$$

Foarte foarte A $$A=\{x, \mu_A^4(x)\} \tag{2.22}$$

Mai puțin A $$Dil(A) = \sqrt{A} = \{x, \sqrt{\mu_A(x)}\} \tag{2.23}$$

Plus A $$A=\{x, \mu_A^{1.25}(x)\} \tag{2.24}$$

Minus A $$A=\{x, \mu_A^{0.75}(x)\} \tag{2.25}$$

Tabelul 2.2 prezintă exemple numerice a operațiilor cu mulțimi fuzzy definite anterior.

**Tabelul 2.2.** Exemple numerice a operațiilor cu mulțimi fuzzy

| **Operația** | **Exemplu numeric** |
|---|---|
| Egalitatea | A={(a1, 0,7), (a2, 1), (a3, 0,4)} egală cu<br>B={(b1, 0,7), (b2, 1), (b3, 0,4)}, |

| | |
|---|---|
| | unde a1=b1, a2=b2, a3=b3 |
| Submulțime | A= {(a1, 0,7), (a2, 1), (a3, 0,4)}, cu submulțimea<br>B={(b1, 0,6), (b2, 0,9), (b3, 0,2)},<br>a1=b1, a2=b2, a3=b3 |
| Intersecție | A={(a1, 0,7), (a2, 1), (a3, 0,4)}<br>B={(b1, 0,2), (a2, 1), (a3, 0,6)},<br>a1=b1, a2=b2, a3=b3<br>Intersecția C={(c1, 0,2), (c2, 1), (c3, 0,4)} |
| Reuniune | A={(a1, 0,7), (a2, 1), (a3, 0,4)}<br>B={(b1, 0,2), (a2, 1), (a3, 0,6)},<br>a1=b1, a2=b2, a3=b3<br>Reuniune C={(c1, 0,7), (c2, 1), (c3, 0,6)} |
| Complementaritate | A={(a1, 0,7), (a2, 1), (a3, 0,4)}<br>Complementara C={(c1, 0,3), (c2, 0), (c3, 0,6)} |
| Sumă algebrică | A={(a1, 0,7), (a2, 1), (a3, 0,4)}<br>B={(b1, 0,2), (a2, 1), (a3, 0,6)},<br>a1=b1, a2=b2, a3=b3<br>Suma algebrică C={(c1, 0,76), (c2, 1), (c3, 0,76)} |
| Produs algebric | A={(a1, 0,7), (a2, 1), (a3, 0,4)}<br>B={(b1, 0,2), (a2, 1), (a3, 0,6)},<br>Produs algebric C={(c1, 0,14), (c2, 1), (c3, 0,24)} |
| Sumă mărginită | A={(a1, 0,7), (a2, 1), (a3, 0,4)}<br>B={(b1, 0,2), (a2, 1), (a3, 0,6)},<br>Suma mărginită C={(c1, 0,9), (c2, 1), (c3, 1)} |

| | |
|---|---|
| Diferență mărginită | A={(a1, 0,7), (a2, 1), (a3, 0,4)}<br>B={(b1, 0,2), (a2, 1), (a3, 0,6)},<br>Diferența mărginită C={(c1, 0,5), (c2, 0), (c3, 0)} |
| Normalizare | A={(a1, 0,5), (a2, 0,9), (a3, 0,2)}<br>Normalizare A={(a1, 0,55), (a2, 1), (a3, 0,22)} |
| Concentrare | A={(a1, 0,5), (a2, 0,9), (a3, 0,2)}<br>Concentrare A={(a1, 0,25), (a2, 0,81), (a3, 0,04)} |
| Dilatare | A={(a1, 0,5), (a2, 0,9), (a3, 0,2)}<br>Dilatare A={(a1, 0,7), (a2, 0,94), (a3, 0,44)} |

A, B, C – mulțimi fuzzy

### 2.4. Întrebări și exerciții

1. Diferența principală dintre o mulțime fuzzy și una clasică constă în faptul că elementele mulțimii fuzzy au atașate funcții de apartenență.
   Adevărat / Fals

2. O mulțime fuzzy poate fi obținută dintr-o mulțime clasică.
   Adevărat / Fals

3. Operațiile care se folosesc pentru manipularea mulțimilor fuzzy sunt exact la fel ca cele folosite în teoria mulțimilor clasice.
   Adevărat / Fals

4. Exercițiu cu operații cu mulțimi fuzzy

Se consideră mulțimea fuzzy A de la pagina 30. Să se determine o submulțime, complementare, normalizarea, concentrarea și dilatarea

# 3

# Logica fuzzy. Numere fuzzy

Acest capitol descriu în amănunt numerele fuzzy, insistându-se pe numere fuzzy triunghiulare și trapezoidale. De asemenea, acest capitol cuprinde și operațiile asociate numerelor fuzzy, și metodele de comparare a numerelor fuzzy. La finalul capitolului sunt introduse exerciții și întrebări care au rolul de ajuta în aprofundarea informațiilor prezentate.

## 3.1 Aspecte generale

Elementele de bază în logica fuzzy sunt mulțimile fuzzy, care sunt definite ca fiind colecții de elemente a căror incluziune în mulțime este caracterizată prin funcții de apartenență, cu valori între 0 și 1, contrar mulțimilor clasice în care elementele sunt caracterizate prin relații de apartenență cu valori exacte 0 și 1.

În inginerie, elementele de bază sunt numerele fuzzy, care în esență sunt mulțimi fuzzy care îndeplinesc anumite condiții. În cele ce urmează sunt definite numere fuzzy, modul de reprezentare a lor și caracteristicile de bază ale acestora.

Un număr fuzzy este o mulțime fuzzy definită pe un interval de încredere din mulțimea numerelor reale, **R**, a cărui funcție caracteristică poate lua orice valoare între 0 și 1. În legătură cu definirea intervalului de încredere, care în literatura de specialitate este definit și ca interval de confidenţă, acesta este o submulţime în **R** care reprezintă un anumit tip de incertitudine. Astfel, un număr fuzzy **A** definit pe intervalul de încredere între $a_1$ și $a_2$, care sunt numere reale se explică ca: **A** nu poate fi mai mic decât $a_1$, dar nici mai mare decât $a_2$.

Reprezentarea simbolică a unui interval de confidență are uzual forma:

$A = [a_1, a_2] = \{x \mid a_1 \leq x \leq a_2\}$.

În general, numerele $a_1$ şi $a_2$ sunt numere distincte și finite. Însă, în anumite cazuri cele două numere pot avea valori speciale, de exemplu:

$a_1 = -\infty$, și / sau $a_2 = \infty$.

În unele cazuri speciale, un număr fuzzy este definit ca singleton, mai exact dacă *x* este un singleton, el poate fi reprezentat ca un interval de forma *x = [x, x]*.

Un număr fuzzy este caracterizată ca fiind o submulțime fuzzy în **R** care are două calități definitoriu: normalitatea și convexitatea.

***Normalitatea*** implică faptul că există cel puțin o valoare în intervalul de încredere a cărei funcție de apartenență are valoarea 1. Acest aspect se poate observa în relația (3.1).

$$\exists x \in R: \forall \mu_A(x) = 1 \tag{3.1}$$

***Convexitatea*** presupune faptul că pentru orice tăietură (tăietura reprezintă o paralelă la axa 0x, care are valoarea funcției de apartenență ca fiind între 0 și 1) cu o dreaptă $\alpha$ paralelă cu axa absciselor 0x. Această caracteristică se observă foarte clar în fig. 3.1. (tăietura la valoarea 0,5 a funcției de apartenență, respectiv 0,8), precum și în relațiile (3.2) și (3.6).

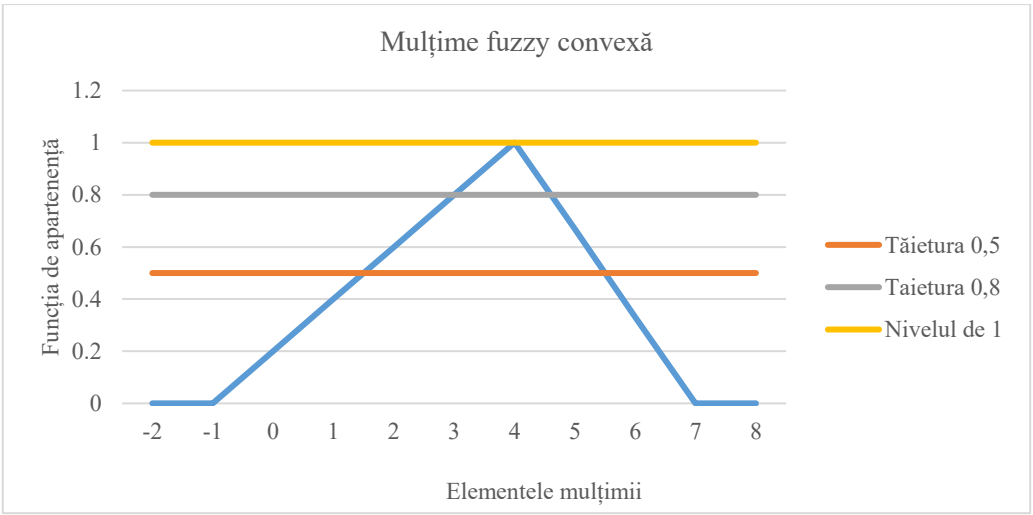



**Fig. 3.1.** Convexitatea unui număr fuzzy

$$A_\alpha = \left[a_1{}^{(\alpha)}, a_3{}^{(\alpha)}\right], \text{ adică } A_{0,5} = \left[a_1{}^{(0,5)}, a_3{}^{(0,5)}\right] \tag{3.2}$$

$$\alpha' < \alpha \Rightarrow \left(a_1{}^{(\alpha')} \leq a_1{}^{(\alpha)}, a_3{}^{(\alpha')} \leq a_3{}^{(\alpha)}\right) \tag{3.3}$$

Reciproc, dacă sunt date două tăieturi

$$A_\alpha = \left[a_1{}^{(\alpha)}, a_3{}^{(\alpha)}\right] \tag{3.4}$$

$$A_{\alpha'} = \left[a_1{}^{(\alpha')}, a_3{}^{(\alpha'')}\right] \tag{3.5}$$

Proprietatea de convexitate implică

$$(\alpha' < \alpha) \Rightarrow \left(A_\alpha \subset A_{\alpha'}\right) \tag{3.6}$$

Alte proprietăţi dorite pentru numerele fuzzy sunt monotonia şi simetria.

Monotonia se referă la caracteristica unui număr fuzzy, prin care valoarea funcției de apartenență cu cât se apropie de 1, cu atât valoarea din intervalul de încredere (axa 0x) este mai aproape de $a_2$ ($\mu_A(x)$ să fie cu atât mai aproape de 1, cu cât *x* este mai aproape de $a_2$).

Simetria poate fi explicată prin relația (3.7), care ne arată faptul că funcția de apartenență ia forma unei forme geometrice simetrice, precum un triunghi isoscel, trapez isoscel etc.

$$\mu_A(a_2 - k) = \mu_A(a_2 + k) \tag{3.7}$$

Despre un număr fuzzy se poate specifica că este un număr fuzzy negativ, respectiv pozitiv, dacă toate valorile intervalului de încredere pentru care funcția de apartenență are valori nenule sunt mai mici ca zero, respectiv mai mari ca zero (relația (3.8)).

$$\mu_A(x) = 0, \quad \forall x < 0 \ (\forall x > 0) \tag{3.8}$$

În literatura de specialitate apar mai multe clasificări în ceea ce privește numerele negative / pozitive, acestea fiind clasificate ca complet negative / pozitive, parțial negative / pozitive, respectiv puțin negative / pozitive.

În cazul numerelor fuzzy complet negative / pozitive, acestea corespund definiției prezentate anterior. Numerele parțial negative / pozitive sunt considerate acele numere fuzzy pentru care nucleu este poziționat în partea negativă / pozitivă a mulțimii **R**, iar cele catalogate ca puțin negative / pozitive au nucleu în afara intervalului negativ / pozitiv, ci doar o parte a intervalului de încredere sunt în partea negativă / pozitivă a **R**.

Reprezentarea generală a numerelor fuzzy se face prin intermediul intervalului încredere, adică valorilor funcției de apartenență care definește suportul numărului fuzzy, $A = [a_1, a_n]$ (în general $n = 3$).

### 3.2 Clase de numere fuzzy

În teoria logicii fuzzy, există o infinitate de tipuri de numere fuzzy însă în aplicaţiile practice se utilizează câteva tipuri speciale de numere fuzzy, care sunt

împărțite în funcție de forma geometrică a funcțiilor de apartenență atașate numerelor fuzzy. Șapte dintre aceste clase speciale, care vor fi descrise în detaliu în acest subcapitol sunt:

- Numere fuzzy triunghiulare (NFT),
- Numere fuzzy trapezoidale (NFTr),
- Numere fuzzy sigmoidale,
- Numere fuzzy în formă de clopot,
- Numere fuzzy exponențiale,
- Numere fuzzy cosinusoidale cu flancuri simetrice,
- Numere fuzzy cosinusoidale cu flancuri nesimetrice.

***Numere fuzzy triunghiulare***

Un număr fuzzy triunghiular poate fi definit ca un triplet ($a_1$, $a_2$, $a_3$), unde $a_1$ și $a_3$ sunt punctele care definesc suportul numărului, iar $a_2$ reprezintă nucleul. Numărul fuzzy triunghiular are funcția caracteristică definită prin expresia matematică din (3.9), iar reprezentarea grafică în fig. 3.2.

$$\mu_A(x) = \begin{cases} 0, x < a_1 \\ \dfrac{x - a_1}{a_2 - a_1}, a_1 \le x \le a_2 \\ \dfrac{a_3 - x}{a_3 - a_2}, a_2 \le x \le a_3 \\ 0, a_3 < x \end{cases} \tag{3.9}$$

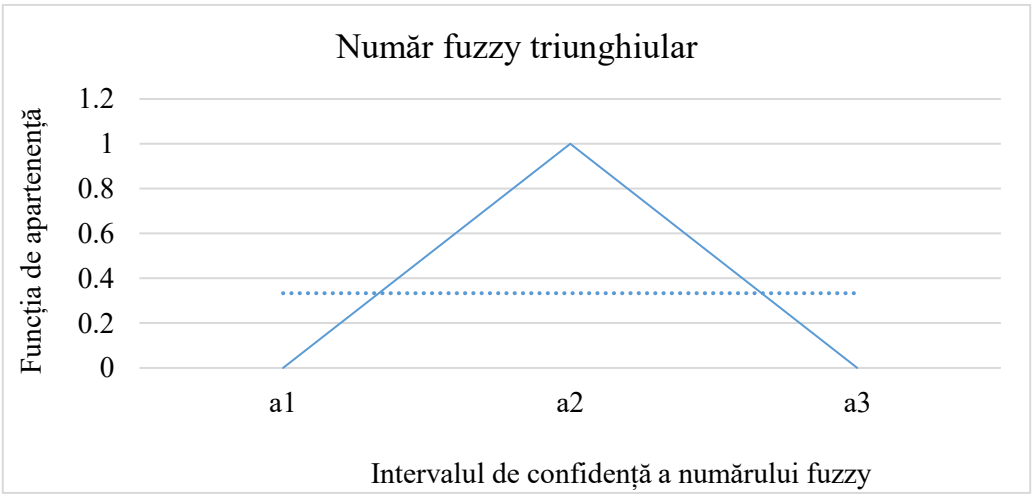



**Fig. 3.2.** Reprezentare grafică - număr fuzzy triunghiular

***Numere fuzzy trapezoidale***

Un număr fuzzy trapezoidal poate fi definit prin patru numere $A = (a_1, a_2, a_3, a_4)$, unde $a_1$ și $a_4$ definesc suportul numărului fuzzy, iar $a_2$ și $a_3$ reprezintă nucleu. Se poate observa că un număr fuzzy triunghiular este o variantă particulară a unui număr trapezoidal, mai exact, un număr trapezoidal pentru care $a_2=a_3$, este un număr fuzzy triunghiular. Reprezentarea analitică a unui număr trapezoidal oarecare, respectiv reprezentarea grafică sunt descrise prin relația (3.10), respectiv fig. 3.3.

$$\mu_A(x) = \begin{cases} 0, x < a_1 \\ \dfrac{x - a_1}{a_2 - a_1}, a_1 \leq x \leq a_2 \\ 1, a_2 \leq x \leq a_3 \\ \dfrac{a_4 - x}{a_4 - a_3}, a_3 \leq x \leq a_4 \\ 0, a_4 < x \end{cases} \tag{3.10}$$

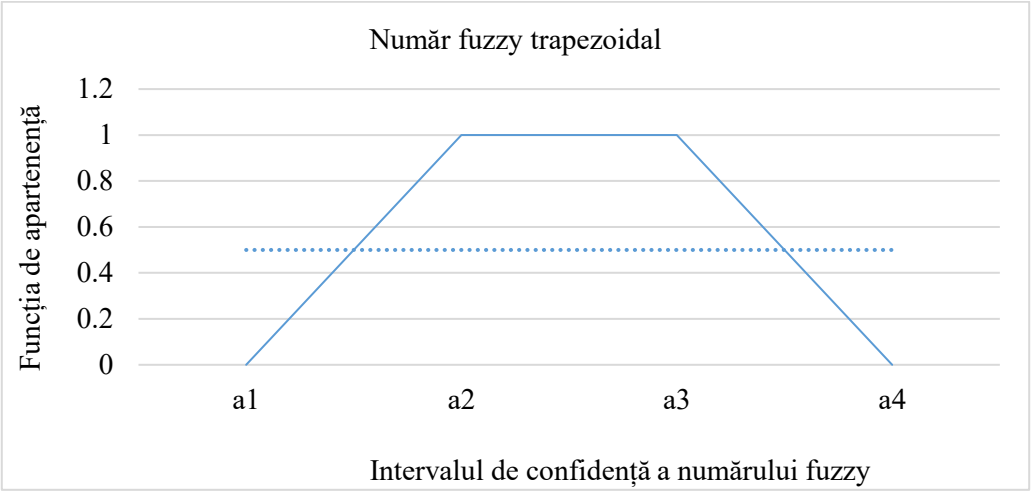



**Fig. 3.3.** Reprezentare grafică - număr fuzzy trapezoidal

***Numere fuzzy sigmoidale***

Un număr fuzzy sigmodidal are funcția de apartenență o funcție sigmoidală definită prin relația matematică (3.11). Particularizând pentru un număr fuzzy, se obține relația (3.12), unde sunt folosite două funcții sigmoidale pentru a realiza diferența dintre ele, și a obține un număr fuzzy complet, care în unele surse de

specialitate este notat cu *Dsigm*. Din relația amintită se observă că numărul fuzzy va avea nucleul în punctul *m*. Reprezentarea grafică din fig. 3.4, ne arată un număr fuzzy sigmodidal particular definit folosind MatLab – Fuzzy Logic Designer. Se observă din grafic, faptul că m = 5, iar parametrii funcțiilor sigmoidale au valorile a1=5, b1=2, a2=5, b2=7.

$$\mu(x) = \frac{1}{1 + e^{-a(x-b)}} \tag{3.11}$$

$$\mu_A(x) = \begin{cases} \dfrac{1}{1 + e^{-a1(x-b1)}}, x \leq m \\ 1 - \dfrac{1}{1 + e^{-a2(x-b2)}}, x > m \end{cases} \tag{3.12}$$

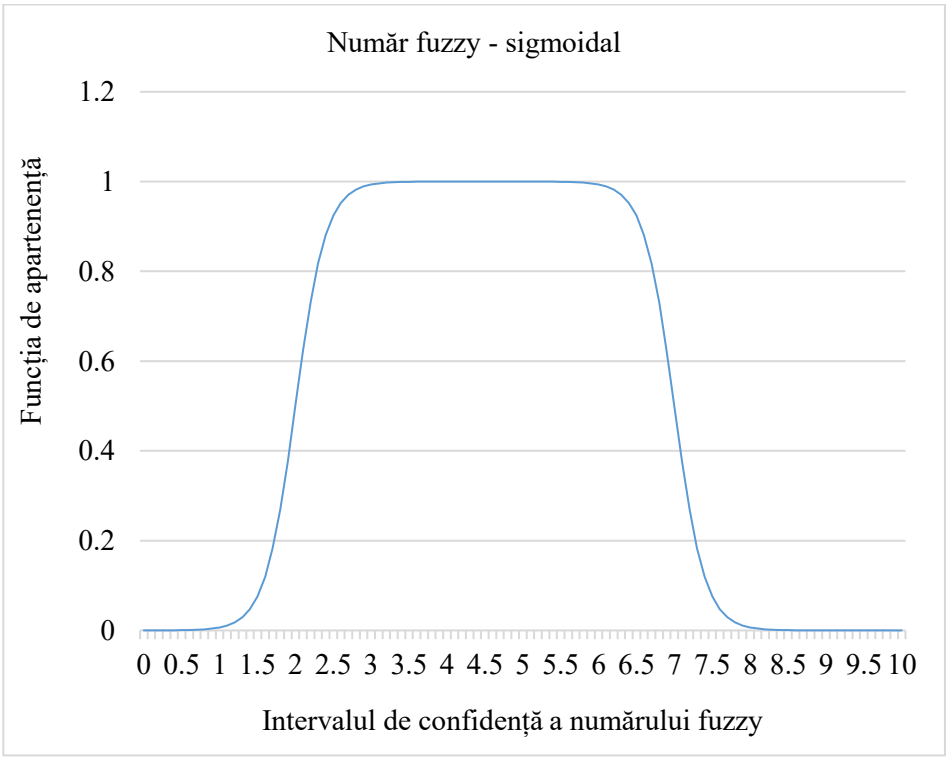



**Fig. 3.4.** Reprezentare grafică - număr fuzzy sigmoidal

***Numere fuzzy în formă de clopot***

Un număr fuzzy în formă de clopot are funcția de apartenență o funcție clopot generalizată definită prin relația matematică (3.13). Reprezentarea grafică din fig. 3.5, ne arată un număr fuzzy clopot particular (gbellmf) definit folosind

MatLab – Fuzzy Logic Designer. Reprezentarea grafică din figură arată un număr fuzzy clopot particular care are parametrii a=2, b=4, iar c=6. Se poate observa din grafic că parametru c indică nucleul numărului fuzzy, a controlează lățimea clopotului, iar b controlează gradul de înclinare a clopotului.

$$\mu_A(x) = \frac{1}{1 + \left|\frac{x-c}{a}\right|^{2b}} \tag{3.13}$$

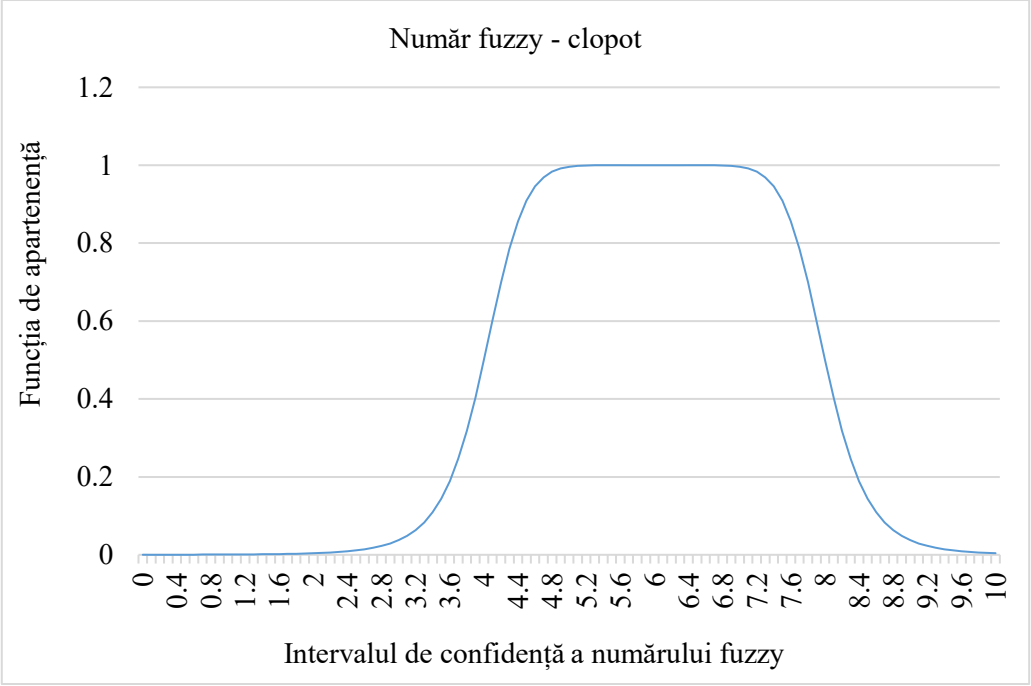



**Fig. 3.5.** Reprezentare grafică - număr fuzzy clopot

***Numere fuzzy Gauss***

Un număr fuzzy Gauss este un caz particular a unui număr fuzzy exponențial. Relațiile (3.14) și (3.15) prezintă forma analitică a unui număr fuzzy exponențial, respectiv a unui număr fuzzy Gauss. Ultimul este folosit des în sistemele fuzzy deoarece are o formă regulată sub formă de clopot. Reprezentarea grafică din fig. 3.6, ne arată un număr fuzzy Gauss definit folosind MatLab – Fuzzy Logic Designer. Reprezentarea grafică din figură arată un număr fuzzy Gauss particular care are parametrii c = 5 (dă valoarea nucleului numărului fuzzy), iar σ=2 (influențează lățimea numărului fuzzy).

$$\mu_A(x) = \begin{cases} e^{-\alpha(c-x)^2}, x \leq c \\ e^{-\beta(x-c)^2}, x > c. \end{cases} \quad (3.14)$$

$$\mu_A(x) = \begin{cases} e^{-\frac{(c-x)^2}{2\sigma^2}}, x \leq c \\ e^{-\frac{(x-c)^2}{2\sigma^2}}, x > c. \end{cases} \quad (3.15)$$

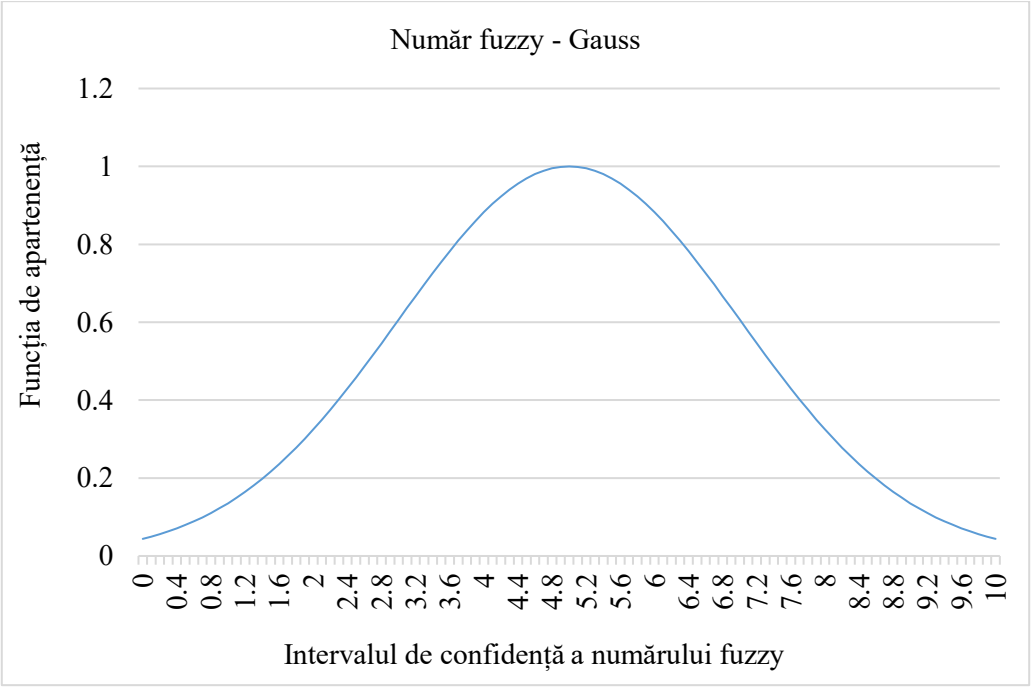



**Fig. 3.6.** Reprezentare grafică - număr fuzzy Gauss

***Numere fuzzy cosinusoidale cu flancuri simetrice***

Un număr fuzzy din clasa numerelor cosinusoidale cu flancuri simetrice are o funcție de apartenență descrisă prin relația (3.16), respectiv grafic în fig. 3.7.

$$\mu_A(x) = \begin{cases} 0, x < x_0 - 2a \\ \frac{1}{2} \cdot \left[1 + cos\left(\frac{\pi \cdot (x - x_0)}{2a}\right)\right], x_0 - 2a \leq x \leq x_0 + 2a \\ 0, x_0 + 2a < x \end{cases} \quad (3.16)$$

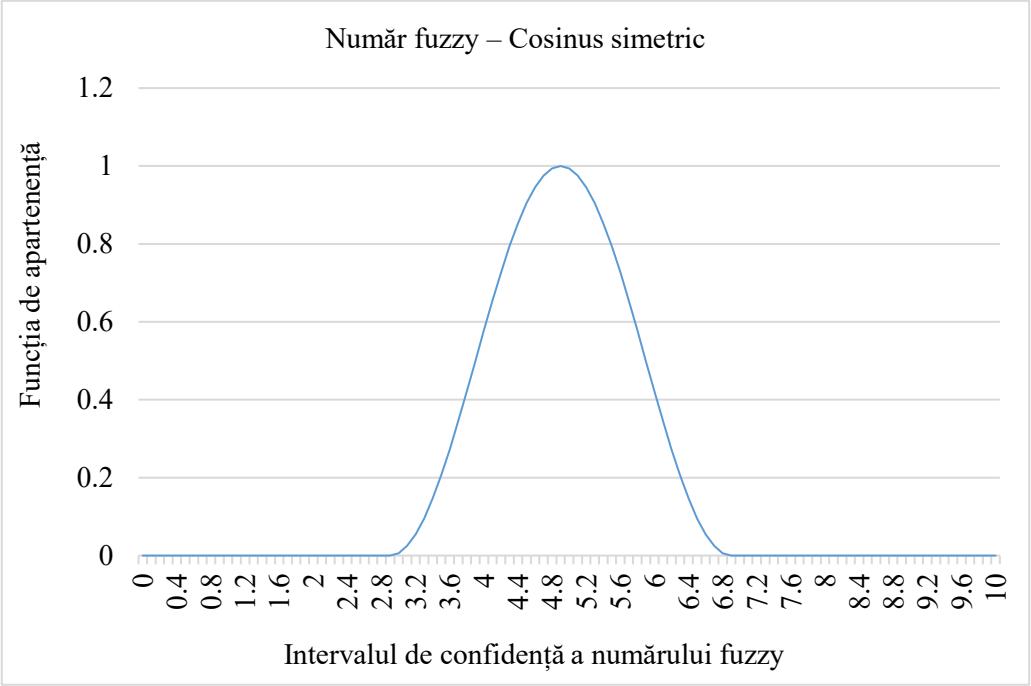



**Fig. 3.7.** Reprezentare grafică - număr fuzzy cosinusoidal simetric

***Numere fuzzy cosinusoidale cu flancuri nesimetrice***

Un număr fuzzy din clasa numerelor cosinusoidale cu flancuri nesimetrice are o funcție de apartenență descrisă prin relația (3.17)ș ele sunt reprezentate printr-o funcție asemănătoare celei din fig. 3.7, cu diferența că flacurile funcției nu sunt simetrice față de nucleul numărului fuzzy.

$$\mu_A(x) = \begin{cases} 0, x < x_1 - 2a_1 \\ \frac{1}{2} \cdot \left[1 + cos\left(\frac{\pi \cdot (x - x_1)}{2a_1}\right)\right], x_1 - 2a_1 \leq x \leq x_1 + 2a_1 \\ 1, x_1 < x < x_2 \\ \frac{1}{2} \cdot \left[1 + cos\left(\frac{\pi \cdot (x - x_2)}{2a_2}\right)\right], x_2 < x < x_2 + 2a_2 \\ 0, x_2 + 2a_2 < x \end{cases} \quad (3.17)$$

### 3.3.Operații cu numere fuzzy

Operațiile cu numere fuzzy își au originea în operațiile cu mulțimi fuzzy. Astfel, numerele fuzzy pot fi manipulate prin adunare, scădere, înmulțire etc. În plus, operațiile cu numere fuzzy se poate face la nivel de interval în încredere,

dar și prin ajutorul funcțiilor de apartenență. Mai mult, anumite clase de numere fuzzy prezintă caracteristici specifice lor în ceea ce privește realizarea operațiilor necesare. În continuare se vor prezenta principalele operații cu numere fuzzy astefl: într-o primă etapă prin utilizarea intervalelor de încredere, apoi prin implicarea funcțiilor de apartenență, iar în final particularitățile care apar atunci când se lucrează cu numere triunghiulare și trapezoidale.

Se consideră două numere fuzzy A și B, definite astfel

$$A = [a_1, a_3], B = [b_1, b_3]$$

***Adunarea*** două numere fuzzy prin folosirea valorilor suport, duce la obținerea unei noi mulțimi fuzzy care are ca suport sumele valorilor suport a celor două numere fuzzy, așa cum se observă în relația (3.18).

$$A(+)B \ = \ [a_1, a_3](+)[b_1, b_3] = [a_1 + b_1, a_3 + b_3] \tag{3.18}$$

***Scăderea*** două numere fuzzy presupune realizarea operație de adunare dintre decăzut și opusul scăzătorului, așa cum arată relația (3.19). Rezultatul va fi un nou număr fuzzy a cărui suport se obține conform relației amintite.

$$A(-)B \ = \ [a_1, a_3](-)[b_1, b_3] = [a_1 - b_3, a_3 - b_1] \tag{3.19}$$

***Opusul*** (imaginea) unui număr fuzzy $A$ se obține prin folosirea valorilor negative a valorilor suport a numărului fuzzy, precum prezintă relația următoare.

$$\bar{A} = [-a_3, -a_1] \tag{3.20}$$

Se observă că în cazul numerelor fuzzy, contrar numerelor singulare, adunarea unui număr fuzzy cu opusul lui nu este 0. Acest aspect este demonstrat prin relația (3.21).

$$A \ (+) \ \bar{A} \ = \ [a_1, a_3] \ (+) \ [-a_3, -a_1] \ = \ [a_1 \ - \ a_3, a_3 \ - \ a_1] \neq 0 \tag{3.21}$$

***Înmulţirea*** a două numere fuzzy prin utilizarea valorilor suport duce la obținerea unui nou număr fuzzy, care are ca suport de stânga valoarea minimă dintre produsul suporturilor, respectiv suport de dreapta valoarea maximă. Relația de mai jos prezintă analitic operația.

$$A(\cdot)B \ = \ [a_1, a_3](\cdot)[b_1, b_3] = [a_1 \cdot b_1 \wedge a_1 \cdot b_3 \wedge a_3 \cdot b_1 \wedge a_3 \cdot b_3, \ a_1 \cdot b_1 \vee a_1 \cdot b_3 \vee a_3 \cdot b_1 \vee a_3 \cdot b_3]; \tag{3.22}$$

***Inversarea*** unui număr fuzzy conduce la un alt număr fuzzy a cărui suport are valorile din relația (3.23), adică suportul de stânga minimul dintre inversul valorilor suportului, iar suportul de dreapta maximul.

$$A^{-1} = [a_1, a_3]^{-1} = \left[\frac{1}{a_1} \wedge \frac{1}{a_3}, \frac{1}{a_1} \vee \frac{1}{a_3}\right] \tag{3.23}$$

***Împărţirea*** a celor două numere fuzzy, presupune mai întâi realizarea împărțirii valorilor suport a deîmpărțitului la valorilor împărțitorului, iar apoi determinarea valorilor minime, respectiv maxime dintre produsele obținute. Aceste calcule sunt descrise în relația următoare.

$$\begin{aligned} A(:)B &= [a_1, a_3]:[b_1, b_3] \\ &= \left[\frac{a_1}{b_1} \wedge \frac{a_1}{b_3} \wedge \frac{a_3}{b_1} \wedge \frac{a_3}{b_3}, \frac{a_1}{b_1} \vee \frac{a_1}{b_3} \vee \frac{a_3}{b_1} \vee \frac{a_3}{b_3}\right] \quad , \end{aligned} \tag{3.24}$$

***Înmulţirea şi împărţirea cu un scalar nenegativ*** a unui număr fuzzy se realizează conform relațiilor (3.25). Aceste operații sunt simplu de realizat, și presupun înmulțirea, respectiv împărțirea valorilor suport la scalarul nenegativ.

$$\begin{aligned} k \cdot A &= [k, k](\cdot)[a_1, a_3] = [ka_1, ka_3] \\ A(:)k &= [a_1, a_3](:)[1/k, 1/k] = [a_1/k, a_3/k] \end{aligned} \tag{3.25}$$

***Minimul*** dintre două numere fuzzy poate rezulta într-un alt număr fuzzy, conform relației (3.26), care este diferit de cele două numere fuzzy considerate, și se obține prin determinarea minimelor valorilor suport.

$$[a_1, a_3] \wedge [b_1, b_3] = [a_1 \wedge b_1, a_3 \wedge b_3]; \tag{3.26}$$

***Maximul*** dintre două numere fuzzy poate rezulta într-un alt număr fuzzy, conform relației (3.27), care este diferit de cele două numere fuzzy considerate, și se obține prin determinarea maximele valorilor suport.

$$[a_1, a_3] \vee [b_1, b_3] = [a_1 \vee b_1, a_3 \vee b_3]; \tag{3.27}$$

***Înmulţirea cu un număr real*** a unui număr fuzzy A rezultă într-un nou număr fuzzy, conform relație (3.28), care are drept suport valoarea minimă dintre produsele prezentate, respectiv valoarea maximă a produselor.

$$A(\cdot)k = [a_1, a_3](\cdot)[k, k] = [a_1 \cdot k \wedge a_3 \cdot k, [a_1 \cdot k \vee a_3 \cdot k] \tag{3.28}$$

Relaţiile anterioare presupuneau folosirea invervalelor de încredere ale numerelor fuzzy, dar așa cum s-a precizat anterior în acest subcapitol, aceleași relaţii se pot implementa şi în cazul numerelor fuzzy reprezentate prin tăieturi.

Tăietura unui număr fuzzy triunghiular, respectiv trapezoidal se determină astfel:

$$\forall \alpha \in [0,1]: A_\alpha = \left[a_1^{(\alpha)}, a_3^{(\alpha)}\right] = [(a_2 - a_1) \cdot \alpha + a_1, -(a_3 - a_2) \cdot \alpha + a_3] \quad (3.29)$$

$$\forall \alpha \in [0,1]: A_\alpha = [(a_2 - a_1) \cdot \alpha + a_1, -(a_4 - a_3) \cdot \alpha + a_4] \quad (3.30)$$

Operaţiile cu numere fuzzy se pot efectua şi folosind funcţiile de apartenenţă. Relațiile sunt enumerate în tabelul 3.1, unde operațiile pot fi explicate astfel:

- Adunarea - pentru fiecare sumă posibilă, $z=x+y$, valoarea funcției de apartenență este determinată prin găsirea celei mai mari valori de apartenență dintre toate perechile (x,y) posibile care însumează z, folosind minimul funcției de apartenență individuale.
- Scăderea - pentru fiecare diferență posibilă $z=x-y$, valoarea funcției de apartenență este determinată prin găsirea celei mai mari valori de apartenență dintre toate perechile (x,y) posibile care satisfac această ecuație, folosind minimul funcției de apartenență individuale.
- Pentru celelalte operații reționamentul este asemănător ca în cazul operație de adunare și scădere.

**Tabelul 3.1.** Operații cu numere fuzzy folosind funcții de apartenență

| **Operația** | **Relația matematică** | **Ecuația** |
|---|---|---|
| Adunarea | $\mu_{A+B}(z) = \bigvee_{z=x+y} (\mu_A(x) \wedge \mu_B(x))$ | (3.31) |
| Scăderea | $\mu_{A-B}(z) = \bigvee_{z=x-y} (\mu_A(x) \wedge \mu_B(x))$ | (3.32) |
| Înmulțirea | $\mu_{A \cdot B}(z) = \bigvee_{z=x \cdot y} (\mu_A(x) \wedge \mu_B(x))$ | (3.33) |
| Impărțirea | $\mu_{A/B}(z) = \bigvee_{z=x/y} (\mu_A(x) \wedge \mu_B(x))$ | (3.34) |
| Minimul | $\mu_{A \wedge B}(z) = \bigvee_{z=x \wedge y} (\mu_A(x) \wedge \mu_B(x))$ | (3.35) |
| Maximul | $\mu_{A \vee B}(z) = \bigvee_{z=x \vee y} (\mu_A(x) \wedge \mu_B(x))$ | (3.36) |

Operaţiile descrise anterior au forme particulare pentru numerele fuzzy triunghiulare şi trapezoidale, iar în paragrafele următoarea se exemplifică acestă particularitate.

Se consideră două numere fuzzy A și B, triunghiulare, respectiv trapezoidale, care sunt definite astfel: A=($a_1$, $a_2$, $a_3$) / A=($a_1$, $a_2$, $a_3$, $a_4$) și B=($b_1$, $b_2$, $b_3$) / B=($b_1$, $b_2$, $b_3$, $b_4$).

Adunarea

$$A + B = (a_1 + b_1, a_2 + b_2, a_3 + b_3)$$
$$A + B = (a_1 + b_1, a_2 + b_2, a_3 + b_3, a_4 + b_4) \quad (3.37)$$

Scăderea

$$A - B = (a_1 - b_3, a_2 - b_2, a_3 - b_1)$$
$$A - B = (a_1 - b_4, a_2 - b_3, a_3 - b_2, a_4 - b_1) \quad (3.38)$$

Simetricul

$$-A = (-a_3, -a_2, -a_1)$$
$$-A = (-a_4, -a_3, -a_2, -a_1) \quad (3.39)$$

Înmulţirea, împărţirea şi inversul se calculează folosind intervale de încredere.

La finalul prezentării operațiilor cu numere fuzy, și în urma unei analize a acestora se poate observa că operațiile descrise anterior se pot aplica pe numere fuzzy triunghiulare şi trapezoidale, iar despre acestea se pot specifica următoarele:

- Adunarea şi scăderea dau un număr fuzzy triunghiular, respectiv trapezoidal;
- Înmulţirea şi împărţirea nu dau în mod necesar un număr fuzzy triunghiular, respectiv trapezoidal;
- Minimul şi maximul nu dau în mod necesar un număr fuzzy triunghiular, respectiv trapezoidal.

***Exemplu numeric***

Date de intrare: A=(-3,2,4) şi B=(-1,0,5) – două numere fuzzy triunghiulare.
Realizarea operațiilor de adunare, scădere și înmuțire au rezultatele
Adunare
A+B=(-3,2,4)+(-1,0,5)=(-4,2,9)
Scădere

A-B= (-3,2,4)-(-1,0,5)=(-3-5,2-0,4-(-1))=(-8,2,5)
Înmulțire
$A_\alpha$=[5α-3, -2 α+4], $B_\alpha$=[α-1,-5 α+5] - tăieturi
$A_\alpha$* $B_\alpha$= [5α-3, -2 α+4]*[α-1,-5 α+5]

### 3.4.Compararea numerelor fuzzy

Compararea numerelor fuzzy este o operație esențială în realizarea și funcționarea sistemelor fuzzy. Aplicații cheie care presupun compararea numerelor fuzzy sunt:

- Luarea deciziilor - În scenariile în care datele exacte nu sunt disponibile, numerele fuzzy pot reprezenta incertitudinea valorilor (de exemplu, cost, timp sau risc). Compararea numerelor fuzzy ajută în alegerea celei mai bune alternative pe baza unor informații vagi sau incomplete.
- Probleme de optimizare – Comparația fuzzy este utilizată în problemele de optimizare cu mai multe obiective, unde obiectivele pot avea constrângeri imprecise, cum ar fi în optimizarea sistemelor de energie (de exemplu, dispecerare economică sau angajamentul unității).
- Evaluare a riscurilor - În inginerie , numerele fuzzy reprezintă riscuri incerte sau rezultate probabilistice. Compararea acestor numere fuzzy ajută la evaluarea riscurilor care sunt mai critice sau mai gestionabile.
- Sisteme de control – Logica fuzzy este utilizată pe scară largă în sistemele de control. Compararea numerelor fuzzy poate face parte din evaluarea performanței sistemului sau ajustarea parametrilor pentru a optimiza procesul de control.
- Proiectare - În proiectarea structurală, de exemplu, materialele ar putea avea proprietăți incerte din cauza variabilității proceselor de fabricație. Compararea numerelor fuzzy ajută la determinarea celor mai potrivite materiale și design.
- Controlul calității - Numerele fuzzy pot reprezenta măsurători sau toleranțe în procesele de control al calității, iar compararea acestora ajută la determinarea dacă produsele îndeplinesc anumite standarde sau nu.
- Prognoza și predicție - numerele fuzzy sunt adesea folosite pentru a modela prognozele incomplete. Compararea predicțiilor neclare permite o planificare mai bună sau o alocare a resurselor.

Aceste aplicații implică manipularea și compararea datelor imprecise, ceea ce face din comparațiile de numere fuzzy un instrument puternic în sistemele energetice reale care sunt caracterizate prin date incerte, incomplete, imprecise.

Compararea numerelor fuzzy se poate realiza prin mai multe tehnici și operatori care sunt descrise în continuare.

Determinarea **depărtării** numărului fuzzy, A, faţă de un număr real, k, este descrisă analitic prin relația (3.40), iar grafic prin fig. 3.8.

$$\text{Rem}(A,k)=R(A,k)=(2A_1+A_2)/2 \tag{3.40}$$

În situația particulară în care numărul fuzzy A este unul triunghiular, respectiv unui trapezoidal, iar depărtarea se calculează față de originea 0, relația (3.41) devine:

$$\begin{aligned} Re\,m\,(A,0) &= \frac{a_1+2a_2+a_3}{4} \\ Re\,m\,(A,0) &= \frac{a_1+a_2+a_3+a_4}{4} \end{aligned} \tag{3.41}$$

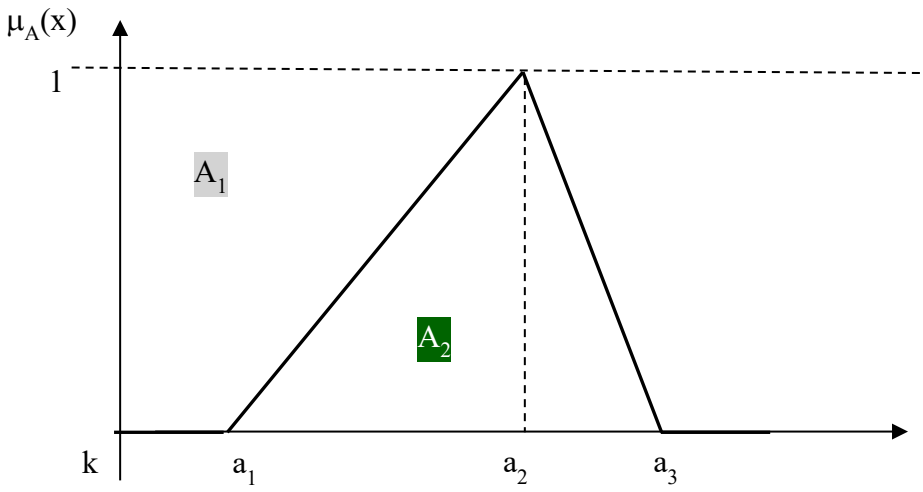



**Fig. 3.8.** Compararea unui număr fuzzy prin depărtarea față de *k*

Un alt criteriu de comparare este criteriul **înălţimii**, h(A). Dacă A are un nucleu, atunci se foloseşte media extremităţiilor nucleului.

Al treia criteriu este criteriul **lățimii**, width(A).

În compararea numerelor fuzzy se aplică întâi criteriul depărtării, dacă valorile sunt egale, atunci se aplică criteriul înălțimii. Dacă nici al doilea criteriu

nu rezultă într-o clasificare clară a numerelor fuzzy, atunci se aplică criteriul lățimii, ca ultimă metodă de clasificare a numerelor.

O altă abordare în compararea și clasificarea numerelor fuzzy este prin folosirea unor funcţii specializate pentru o anumită clasă de numere fuzzy. Pentru numerele fuzzy triunghiulare: Indicele Yager de ordinul 1, indicele Yager de ordinul 3, relaţia lui Adamao şi indicele Promedia.

***Indicele Yager de ordinul 1*** este o valoare caracteristică a unui număr fuzzy care arată poziția numărului fuzzy față de alte numere cu care este comparat. Valoare indicelui Yager de ordinul 1 se calculează prin utilizarea relație (3.42)

$$Y_1(A) = \int_0^1 F_A(\alpha) d\alpha \qquad (3.42)$$

Unde $F_A(\alpha) = \frac{A_L(\alpha)+A_U(\alpha)}{2}$ este funcție medie a tăieturii la nivelul α,
iar $A_L$ și $A_U$ sunt valorile suport ale tăieturii, care satisfac condiția
ca să aibă valori mai mari decât nivelul α.

Prin compararea numerelor fuzzy folosind acest indice, numărul cu valoarea mai mare va fi considerat ca fiind numărul fuzzy mai mare.

În cazul numerelor fuzzy triunghiulare, determinarea indicelui Yager de ordinul 1 se rezumă la calcularea valorii medii folosind relația $C(A) = \frac{a1+a2+a3}{2}$. În final compararea a două numere fuzzy prin intermediul indicelui Yager se face prin determinarea distanței dintre valorile medii ale celor două numere.

***Indicele Yager de ordinul 3*** este o extensie a indicelui Yager de ordinul 1, deoarece acest indice nu ia în calcul doar valoarea nucleului, dar și forma și lățimrea numărului fuzzy. Relația matematică pentru determinarea indicelui Yager de ordinul 3 este (3.43). Comparativ cu indicele Yager de ordinul 1, acest indice amplifică diferențele dintre numerele fuzzy, punând accent pe valorile mai mari ale funcțiilor de apartenență.

$$Y_3(A) = \int_0^1 (F_A(\alpha)^3) d\alpha \qquad (3.42)$$

Unde $F_A(\alpha) = \frac{A_L(\alpha)+A_U(\alpha)}{2}$ este funcție medie a tăieturii la nivelul α,
iar $A_L$ și $A_U$ sunt valorile suport ale tăieturii, care satisfac condiția
ca să aibă valori mai mari decât nivelul α.

***Relația lui Adamao*** se referă la compararea numerelor fuzzy prin determinarea unei valori specifice unui număr fuzzy conform relației (3.43).

$$x^* = max\{x|x \in A(\alpha^*)\} \tag{3.43}$$

Unde $A(\alpha^*)$ este tăietura numărului fuzzy la nivelul $\alpha^*$, care este valoarea maximă a funcției de apartenență, iar valorile funcției de apartenență se consideră ca fiind cele mai mari decât $\alpha^*$. În final valoarea căutată este maximul dintre valorile funcției de apartenență.

Prin folosirea relației lui Adamao, se determină o valoare, care în cazul comparării a două numere fuzzy va indica număru fuzzy mai mare, ca fiind acela pentru care valoarea calculată este maximă.

***Indicele Promedia*** este o caracteristică numerică a numerelor fuzzy prin care acestea se poat compara și clasifica. Relația matematică prin care se determină valoarea indicelui Promedia este (3.44).

$$P(A) = \int_0^1 \frac{A_L(\alpha) + 2A_M(\alpha) + A_U(\alpha)}{4} d\alpha \tag{3.44}$$

Unde $A_L$ și $A_U$ sunt valorile suport ale tăieturii, iar $A_M$ este valoarea medie a tăieturii la nivelul $\alpha$, iar valoare medie se calculează ca fiind $A_M(\alpha) = \frac{A_L(\alpha)+A_U(\alpha)}{2}$.

Comparativ cu indicele Yager, indicele Promedia oferă o manieră mai echilibrată de a compara numerele fuzzy, întrucât se folosește și de valoarea medie a numărului fuzzy.

În continuare pentru înțelegearea mai clară a metodelor de comparare menționate sunt date exemple numerice.

***Exemplu numeric***

Se consideră următoarele numere fuzzy triungiulare:

A =(-3, 0, 3).

B =(-2, -1, 2).

C =(0, 1, 2).

D =(-1, 3, 4).

E =(1, 2, 4).

F =(2, 3, 7).

Se clasifică numerele fuzzy A, B, C, D, E, F prin utilizarea metodologiei care cuprinde calculul depărtării, a înălțimii și a lățimii.

Se va calcula depărtarea numerelor față de scalarul -3, valorile se notează Rem(număr fuzzy, -3).

Rem(A,-3)=3, Rem(B,-3)=2, Rem(C,-3)=3, Rem(D,-3)=4, Rem(E,-3)=6, Rem(F,-3)=6

Se calculează înălțimea numerelor fuzzy,

H(A) = 0, H(B)=-1, H(C)=1, H(D)=3, H(E)=2, H(F)=3.

Lățimea numerelor fuzzy este egală cu

W(A) = 6, W(B)=4, H(C)=2, W(D)=5, W(E)=3, W(F)=5.

Confrom depărtării față de -3, apoi a înălțimii și la final a lățimii, numere fuzzy se pot clasifica astfel: B<A<C<D<E<F

Se clasifică numere fuzzy folosind indicele Yager de ordinul 1. Valorile medii ale numerelor fuzzy considerate sunt: Y(A)=0, Y(B)=-1/3, Y(C)=1, Y(D)=2, Y(E)=2,33, Y(F)=4. Se poate observa că, clasificarea este aceeași ca din metodologia procedentă, adică B<A<C<D<E<F.

## 3.5. Întrebări și exerciții

### Întrebări

1. Numerele fuzzy sunt total diferite de mulțimile fuzzy, întrucât cele din urmă au funcții de apartenență neliniare.
   Adevărat / Fals

2. Numerele fuzzy triunghiulare sunt un caz particular a numerelor fuzzy
   Trapezoidale / Gauss / Pătratice

3. Numerele fuzzy triunghiulare au operații specifice lor, care nu se aplică pentru alte numere fuzzy.
   Adevărat / Fals

4. Prin înmulțirea a două numere trapezoidale rezultă un număr fuzzy care nu este trapezoidal.

Adevărat / Fals

### Exerciții

1. Realizaţi operaţii cu numere fuzzy triunghiulare şi trapezoidale.

Se consideră două numere fuzzy triunghiulare și trapezoidale A și B. Să se realizeze operațiile menționate la capitolul 3.2.
A = (-1, 1, 3), B = (3, 5, 7).
A = (-1, 1, 2, 3), B = (3, 4, 5, 7).

2. Ordonaţi numere fuzzy triunghiulare şi trapezoidale.
   A = (-1, 1, 3), B = (3, 5, 7), C = (-1, 2, 4), D = (4, 6, 8).
   A = (-1, 1, 2, 3), B = (3, 4, 5, 7), C = (-1, 2, 3, 4), D = (4, 6, 7, 8).

3. Să se scrie 2 numere fuzzy triunghiulare sau trapezoidale. Să se reprezinte prin intervalul de încredere, tăieturi, funcția de apartenență și grafic. Să se realizeze suma, scăderea, înmulțirea și împărțirea. Pentru noile numere fuzzy se va specifica intervalul de încredere și tăieturile. Să se aleagă o valoare din intervalul de încredere pentru care să se calculeze valoarea funcției de apartenență.

# 4

# Logica fuzzy. Sisteme fuzzy

Acest capitol prezintă sistemele fuzzy în funcție de structură și aplicabilitate. Astfel, sunt descrise în detaliu regulatoarele fuzzy, mai exact structura, caracteristicile și implementarea lor, precum și atributele definitorii ale sistemelor fuzzy generale.

## 4.1. Introducere

Sistemele fuzzy se caracterizează prin faptul că în componenţa lor au elemente care funcţionează folosind logica fuzzy. Aceste elemente pot conține numere fuzzy, operații cu numere / mulțimi fuzzy, sau raționamentul specific logicii fuzzy, sau toate. În consecință, sistemele fuzzy pot să fie sisteme clasice, care conțin doar logica fuzzy, precum este cazul regulatoarelor fuzzy, sau sisteme hibride, care cuprind metode convenționale, logica fuzzy, dar și alte tehnici de IA.

În acest capitol se vor prezenta sistemele fuzzy clasice folosite în control sau alte aplicații, urmând ca sistemele hibride să fie subiectul ultimului capitol, adică capitolul 14. O listă mai detaliată a modului cum este folosită logica fuzzy în ingineria energetică este enumerată în continuare.

- Aritmetica fuzzy și numerele fuzzy ajută la gestionarea incertitudinii în prognoza sarcinii, analiza fluxului de putere și modelarea dinamică a sistemelor de putere.
- Ecuațiile diferențiale fuzzy modelează dinamica frecvenței controlate, reglarea tensiunii și stabilitatea în condiții complexe.
- Optimizarea fuzzy se aplică fluxului optim de putere și managementului multi-obiectiv al energiei în condiții de incertitudine.
- Probabilitatea și statisticile fuzzy îmbunătățesc evaluarea fiabilității, predicția defecțiunilor și analiza riscurilor în sistemele de alimentare.
- Metode numerice precum analiza circulației de putere sunt utilizate pentru a rezolva sisteme cu date imprecise sau incerte.

- Controlul fuzzy este folosit în reglarea proceselor neliniare, care conțin mult zgomot, sau sunt caracterizate de incertitudini.

### 4.2.Controlul fuzzy

Controlul sau reglarea unei mărimi se referă la aducerea şi menţinearea mărimii fizice dintr-un proces tehnologic la o anumită valoare. Mărimea fizică care este reglată se numeşte mărime reglată; ea poate fi presiune, tensiune, curent, temperatură etc.Valoarea la care trebuie adusă mărimea reglată se numeşte mărime de referinţă (prescrisă).

Procesul de control de desfășoară urmând doi pași (1) mărimea reglată este măsurată şi comparată cu mărimea de referinţă și (2) dacă există o diferenţă, atunci aceasta este corectată. Corecția mărimii reglate se poate realiza manual sau automat.

Reglarea automată se caracterizează prin schema de principiu din fig. 4.1.

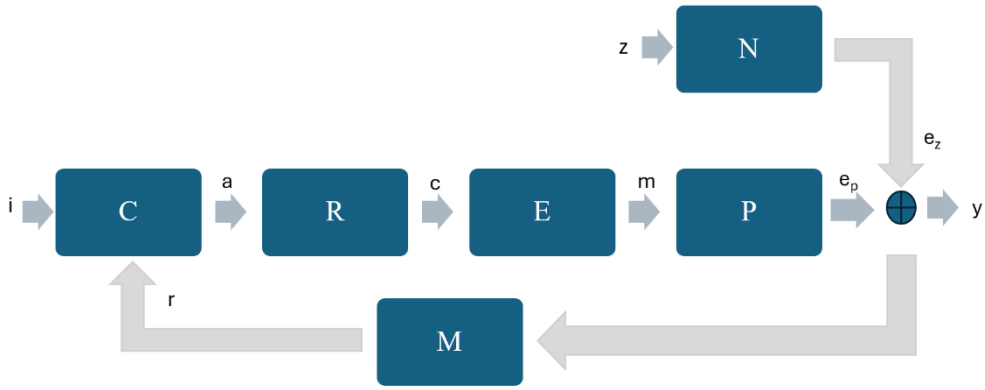



**Fig. 4.1** Schema de principiu a reglării automate. P – procesul, M – elementul de măsurare, C– elementul de comparație, R – regulator, E – elementul de execuție, i-mărime de intrare, y-mărime de ieșire, r- mărimea de reacție, a=i-r mărime de acționare, c- mărime de comandă, m- mărime de execuție, z- perturbație, $e_p$- mărimea de ieșire influențată de proces, $e_z$ -mărimea de ieșire influențată de perturbație

Regulatorul este elementul de bază a sistemului de control, care în reglarea automată clasică poate fi un controler histerezis, proporțional, integrator, derivator, sau varianta cea mai complexă PID (proportional-integrator-derivator). Controlul clasic are anumite neajunsuri, iar în comparație cu acesta, controlul fuzzy oferă mai multe avantaje. Avantajele utilizării controlului fuzzy comparativ cu controlul PID sunt:

- Logica fuzzy gestionează neliniaritatea mai bine decât PID - Controlerele PID funcționează bine pentru sisteme liniare cu dinamică bine definită, dar multe sisteme din energetică sunt foarte neliniare. Logica fuzzy se poate adapta mai bine acestor condiții fără a necesita un model matematic exact.
- Reglarea PID nu este întotdeauna simplă - reglarea PID este dificil de implementat în cazul sistemelor complexe. Logica fuzzy oferă o alternativă atunci când metodele tradiționale de reglare eșuează sau când dinamica sistemului se schimbă frecvent. Poate funcționa bine cu euristică, cunoștințe de specialitate sau strategii de control adaptiv.
- Controlul fuzzy excelează în cazul sistemelor caracterizate prin incertitudini și afectate de zgomote - Spre deosebire de PID, care se bazează în mare măsură pe modele matematice precise, controlerele fuzzy pot gestiona incertitudinea și intrările imprecise. Acest lucru este util în special în sistemele electroenergetice, în care parametrii și factorii fluctuează în mod imprevizibil.

Prin folosirea logicii fuzzy, acest regulator convențional este înlocuit cu un regulator fuzzy, sau un regulator hibrid care cuprinde un regulator convențional îmbunătățit cu logica fuzzy.

Avantajele abordării logicii fuzzy în controlul proceselor vin din abilitatea logicii fuzzy de a:

- Formaliza și simula cunoștințele și experiența unui operator în controlul proceselor și acordarea regulatoarelor.
- Furniza un răspuns relativ simplu pentru procese care sunt relativ greu de modelat.
- Ține în permanență cont de situațiile speciale sau excepții.
- Lua în considerare mai multe variabile și de a considera diferite ponderi ale acțiunii acestora.

În energetică controlul fuzzy se folosește cu succes în:

1. Controlul automat al producerii energiei electrice,
2. Reglarea tensiunii,
3. Stabilizarea sistemului de putere,
4. Integrarea energiei regenerabile,
5. Controlul frecvenței,
6. Fault Ride-Through (FRT) în parcurile eoliene,

7. Managementul energiei în microrețea.

Un regulatorul fuzzy tratează informația utilizând inferențele (inferență = operație logică de trecere de la un enunț la altul și în care ultimul enunț este dedus din primul, operație prin care se ajunge la o anumită concluzie care, deși nu este logic derivabilă din premise) dintre mai multe reguli, bazându-se pe variabile lingvistice, ci nu pe relații și funcții matematice cum este caracteristic controlului convențional. Inferențele sunt tratate utilizând operatori logici, precum ȘI, SAU, NOT etc.

O condiție fundamentală pentru utilizarea și implementarea reglării fuzzy o reprezintă existența unei expertize umane și a unor cunoștințe practice solide și corecte a procesului controlat. În lipsa acestora, dezvoltarea și implementarea unui regulator fuzzy eficace și eficient este destinată eșecului.

Structura unui controler fuzzy cuprinde trei blocuri principale: blocul de fuzzyficare, blocul de reguli, blocul de defuzzyficare. Fig. 4.2. ilustrează grafic componența unui regulator fuzzy.


**Fig.4.2.** Structura unui controler fuzzy

În dezvoltarea unui controler fuzzy trebuie să se respecte următorii pași:

1. Determinarea datelor de intrare și ieșire a controlerului fuzzy
   Datele de intrare sunt mărimile fizice care influențează procesul de control, iar datele de ieșire reprezintă mărimile de execuția în procesul de control.
2. Fuzzyficarea datelor de intrare și ieșire
   Fuzzyficarea se referă la acțiunea prin care se atribuie mărimilor fizice de intrare și ieșire variabile lingvistice, valori lingvistice și numere fuzzy, mai exact, se găsesc corespondenți în logica fuzzy a datelor de intrare și ieșire, cu care va lucra controlerul fuzzy.
3. Scrierea regulilor
   Se scriu toate regulile care arată dependența dintre mărimile de intrare și ieșire.
4. Alegerea metodei de rezolvare a inferențelor

Se alege metoda prin care se vor rezolva inferențele dintre reguli pentru a determina suprafața fuzzy de ieșire.

5. Alegerea metodei de defuzzyficare
   Se alege metoda prin care se reazlizează defuzzyficarea – acțiunea prin care se determină valoarea reală singulară a mărimii de ieșire pornind de la suprafața fuzzy de ieșire.

***Fuzzyficarea datelor de intrare și ieșire***

Fuzzyficarea datelor cu care lucrează regulatorul fuzzy înseamnă că pentru fiecare mărime cu care lucrează regulatorul fuzzy se va defini o variabilă lingvistică, cu valorile lingvistice corespunzătoare și numerele fuzzy asociate lor. Astfel, la ieșirea din acest bloc, datele controlerului vor fi reprezentate prin numere fuzzy, care de cele mai multe ori sunt reprezentate prin vectori.

Procesul prin care se asociază mărimilor fizice variabile lingvistice se numește și modelare lingvistică, care se definește (precum s-a menționat anterior) ca acțiunea prin care unei mărimi fizice i se asociază o variabilă lingvistică având una sau mai multe valori și cărora li se asociază diverse funcții de apartenență. Modelarea lingvistică își are originea în modul de reprezentare a informațiilor și regulilor de către experți umani. Deoarece, descrierea practică a unei anumite situații de către un expert uman, conține în general formulări vagi precum câteva, mult, adesea, rar, cald, frig etc. Expresiile de acest gen pot fi ușor transformate în variabile lingvistice din logica fuzzy.

Fig. 4.3 ilustrează rezultatul grafic a fuzzyficării unei mărimi fizice.

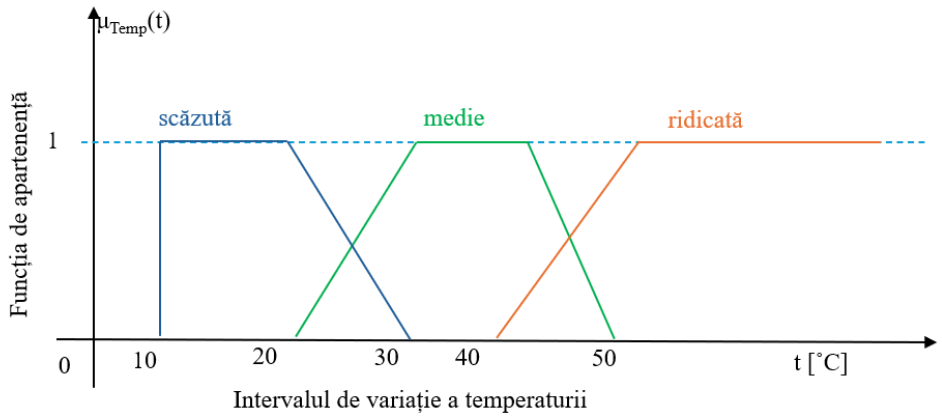



**Fig. 4.3.** Reprezentarea grafică a unei mărimi fuzzyficate

În figură este reprezentată o mărime fizică, căreia îi este asociată variabila lingvistică Temp (temperatura), care are trei valori lingvistice: scăzută, medie și ridicată. Fiecărui valori lingvistice îi este asociat un număr fuzzy trapezoidal. Etapele realizate în obținearea acestei reprezentări grafice au fost:

1. Denumirea variabilei lingvistice.
2. Determinarea intervalului de variație a mărimii. În cazul prezentat este [10, 60].
3. Determinarea numărului de valori lingvistice, denumirea lor și intervale lor de încredere. În fig. 4.3 sunt trei valori ligvistice, având numele și intervalele de încredere scăzută [10, 30], medie [20, 50] și ridicată [40, 60]. Se observă faptul că aceste intervale se suprapun, ducând la apariția unor intervale de incertitudine, ceea ce este caracteristic logicii fuzzy, și în același timp obligatoriu.
4. Alegerea tipului de numere fuzzy asociate valorilor lingvistice și definirea funcțiilor de apartenență ale acestora. În exemplu dat, au fost folosite numere fuzzy trapezoidale. Folosind relația (3.10) relațiile matematice ale funcțiilor matematice ale numerelor fuzzy sunt:

$$\mu_{Temp}^{scăzută}(t) = \begin{cases} 0, & t < 10 \\ \dfrac{30-t}{10}, & 20 \le t \le 30 \\ 1, & 10 \le t \le 20 \\ 0, & 30 < t \end{cases} \tag{4.1}$$

$$\mu_{Temp}^{medie}(t) = \begin{cases} 0, & t < 20 \\ \dfrac{t-20}{10}, & 20 \le t \le 30 \\ 1, & 30 \le t \le 45 \\ \dfrac{50-t}{5}, & 45 \le t \le 50 \\ 0, & 50 < t \end{cases} \tag{4.2}$$

$$\mu_{Temp}^{ridicată}(t) = \begin{cases} 0, & t < 40 \\ \dfrac{t-40}{5}, & 40 \le t \le 55 \\ 1, & 55 \le t \le 60 \\ 0, & 60 < t \end{cases} \tag{4.3}$$

Observații:

1. Numele variabilei lingvistice și a valorilor lingvistice sunt alese subiectiv, în funcție de preferințele specialistului.

2. Numărul de variabile ligvistice și a numerelor fuzzy asociate în literatura de specialitate este impar, cel mai comun fiind 7, 9 sau 11.
3. Tipul numerelor fuzzy se alege în funcție de modul de variație a mărimilor fizice fuzzyficate. În literatura de specialitate cele mai populare clase de numere fuzzy folosite sunt cele triunghiulare și trapezoidale.

***Blocul de reguli (inferențe)***

Variabilele lingvistice atașate unui proces de reglare sunt legate între ele printr-un set de reguli (inferențe)/deducții fuzzy. Aceste inferențe arată modul în care mărimile de ieșire se modifică în funcție de asocierea mărimilor de intrare.

Inferențele se împart în două mari categorii, și anume inferență cu o singură regulă și cu mai multe reguli.

Inferența cu o singură regulă se folosește atunci când rezultatul poate fi obținut printr-o singură regulă de forma

$$u = [x_1 \text{ SAU } (x_2 \text{ ȘI } x_3) \text{ SAU } \ldots.] \text{ ȘI } x_n$$

unde $x_1$ ..... $x_n$ sunt variabilele de intrare, iar u este ieșirea.

Așa cum se poate observa, ieșirea este obținută din variabilele de intrare, care sunt conectate între ele prin operatorii logici ȘI și SAU, forma regulii depinde de natura problemei.

Variabilele de intrare sunt caracterizate printr-o funcție de apartenență, iar valorile numerice sunt diferite pentru fiecare concurent (variabilele de intrare concurează în interiorul regulii singulare).

Această metodă de inferență este simplistă și în multe situații nu are capacitatea de a caracteriza corespunzător procese reale, care sunt mai complexe. În această situație se folosesc inferențe cu mai multe reguli.

Inferența cu mai multe reguli apare atunci când se folosesc mai multe variabile care pot avea diverse valori, iar decizia trebuie luată diferit, în funcție de valorile pe care le au aceste variabile.

Regulile pot fi exprimate sub forma generală

operație = dacă condiție 1, atunci operație 1 SAU
= dacă condiție 2, atunci operație 2 SAU
.....
= dacă condiție n, atunci operație m

Inferențele cu mai multe reguli pot să fie descrise prin mai multe metode: descrierea lingvistică, descriere simbolică, matricea de reguli sau tabelul cu reguli. Indiferent de modul cum sunt descrise regulile, inferențelor cu mai multe reguli este caracteristic faptul că sunt verificate și utilizate mai multe reguli simultan.

Descrierea lingvistică este metoda prin care se prezintă regulile în cel mai descriptiv mod. Exemplul următor subliniază acestă afirmație.

Dacă (x este negativ și y este pozitiv atunci u este negativ) SAU
Dacă (x este zero și y este pozitiv atunci u este pozitiv) SAU ....

Descrierea simbolică este mai compactă decât metoda precedentă, ea se bazează pe folosirea mai mult a simbolurilor, așa cum este evidențiat în exemplul următor.

Dacă (x=N și y=P, atunci u=N) SAU
Dacă (x=Z și y=P, atunci u=P) SAU...

Descrierea regulilor prin folosirea matricei de inferențe este prima dintre abordările compacte. Tabelul 4.1 cuprinde un exemplu de scriere a matricei de inferențe a celor nouă reguli, unde x și y sunt datele de intrare, iar u ieșirea. Valorile lingvistice pentru x sunt N, Z, și P iar pentru y sunt N, Z și P. Ieșirea are valorile lingvistice N, Z, și P.

**Tabelul 4.1.** Matricea de inferențe

| **U** | | **X** | | |
|---|---|---|---|---|
| | | N | Z | P |
| | N | P | N | Z |
| **Y** | Z | P | Z | N |
| | P | N | P | Z |

Descrierea regulilor prin folosirea tabelei de inferență este a doua abordare compactă. Tabelul 4.2. descrie un exemplu de scriere a nouă reguli din tabelul 4.1, dar sub forma tabelei de inferențe.

**Tabelul 4.2.** Tabela de inferențe

| Regula | X | Y | U |
|---|---|---|---|
| 1 | N | N | P |
| 2 | N | Z | P |
| 3 | N | P | N |
| 4 | Z | N | N |
| 5 | Z | Z | Z |
| 6 | Z | P | P |
| 7 | P | N | Z |
| 8 | P | Z | N |
| 9 | P | P | Z |

O comparație între inferențele cu o singură regulă și inferențele cu mai multe reguli scoate în evidență următoarele:

- Inferența cu o regulă este mai simplă și rapidă, dar are o putere expresivă limitată, și funcționează bine atunci când sistemul este simplu și este necesară doar o relație de bază între intrări și ieșiri.
- Inferența cu reguli multiple este mai complexă, dar și flexibilă și robustă. Ea poate descrie relațiile nuanțate dintre intrări și ieșiri, dar necesită mai multe resurse de calcul, deoarece toate regulile trebuie evaluate și rezultatele lor agregate. Totuși, acesta este standardul în majoritatea sistemelor de control fuzzy din lumea reală, datorită capacității de a modela luarea deciziilor complexe.

***Metode de rezolvare a inferențelor***

În cazul inferențelor cu o singură regulă, rezultatul va fi un număr fuzzy singular, care va deveni intrarea pentru blocul de defuzzyficare. Dacă se folosesc inferențe cu mai multe reguli, atunci este necesar să se aplice o metodă de rezolvare a inferențelor. În controlul fuzzy, cele mai populare trei metode de obținere a ieșirii fuzzy sunt: metoda Max-Min, metoda Max-Prod, și metoda Sum-Prod. Fiecare dintre aceste metode are specific modul în care sunt folosiți operatorii logici și cuvintele cheie din interiorul regulilor sau dintre reguli. Aceste

cuvinte cheie sunt ȘI și SAU din interiorul regulilor, dintre variabilele de intrare, Atunci, respectiv SAU dintre reguli. În continuare se descrie modul în care aceste cuvinte sunt folosite în funcție de abordare.

Metoda Max-Min

- Operatorul SAU din regulă este perceput ca maxim – se realizează maximul dintre valorile funcțiilor de apartenență ale variabilelor de intrare, valoare care este transmisă mai departe.
- Operatorul ȘI din regulă este perceput ca minim – se efectuează minimul dintre valorile funcțiilor de apartenență ale variabilelor de intrare; valoarea maximă va fi transmisă mai departe.
- Atunci este văzut ca un ȘI (minim) – se efectuează operația de minim dintre scalarul ales înainte de Atunci și numărul fuzzy a ieșirii.
- SAU dintre reguli este perceput ca maxim – pe intervalul de variație a variabilei de ieșire ($a_1$, $a_n$) se va alege valoarea maximă dintre valorile funcțiilor de apartenență.

Metoda Max-Prod

- Operatorii SAU și SI din reguli sunt percepuți ca maxim și minim – se procedează la fel în cazul metodei Max-Min, adică se realizează maximul / maximul dintre valorile funcțiilor de apartenență a variabilelor de intrare.
- ”Atunci” este văzut ca produs – se realizează produsul dintre scalarul obținut la etapa anterioară și numărul fuzzy al ieșirii.
- SAU dintre reguli este perceput ca maxim - pe intervalul de variație a variabilei de ieșire ($a_1$, $a_n$) se va alege valoarea maximă dintre valorile funcțiilor de apartenență.

Metoda Sum-Prod

- Operatorul SAU din regulă este perceput ca sumă – se realizează suma dintre valorile funcțiilor de apartenență a variabilelor de intrare. Această sumă este transmisă mai departe, ajungând la operatorul ”Atunci”.
- Operatorul ȘI din regulă este perceput ca produs - se realizează produsul dintre valorile funcțiilor de apartenență a variabilelor de intrare. Acest produs este transmis mai departe, ajungând la operatorul ”Atunci”.

- ”Atunci” este văzut ca un produs - se realizează produsul dintre scalarul obținut la etapa anterioară și numărul fuzzy al ieșirii.
- SAU dintre reguli este perceput ca sumă - pe intervalul de variație a variabilei de ieșire ($a_1$, $a_n$) se va realiza suma dintre valorile funcțiilor de apartenență a numerelor fuzzy de la ieșire.

După aplicarea regulilor / a regulii inferenței rezultă un mulțime fuzzy complexe, care va deveni intrarea pentru blocul de defuzzyficare.

***Blocul de defuzzyficare***

Pentru obținerea ca rezultat a unui singur număr real trebuie efectuată o operație care să țină cont de diferite condiții – defuzzyficare. Deci, sarcina operației de defuzzyficare este de a găsi o singură valoare reală care să caracterizeze mulțimea fuzzy rezultată la ieșire. În practică există multe metode de defuzzyficare, dintre care cele mai folosite sunt:

- Metoda centrului de greutate,
- Metoda centrului sumelor,
- Metoda bisectoarei,
- Metoda mediei ponderate,
- Metoda înălțimii,
- Metoda centrului suprafeței celei mai mari,
- Metoda primului/ultimului dintre maxime,
- Metoda maximului din mijloc.

Metoda centrului de greutate

Metoda centrului de greutate este o metodă populară printre specialiști, și presupune determinarea abscisei centrului de greutate a suprafeței aflată sub funcția de apartenență combinată. Relația de calcul în cazul în care funcția de apartenență este discretă este (4.4), iar în situația unei funcții de apartenență continue (4.5). Explicarea grafică a acestei metode este ilustrată în fig. 4.4.

$$u^* = \frac{\sum_{i=1}^{l} u_i \cdot \mu_{Bu}(u_i)}{\sum_{i=1}^{l} \mu_{Bu}(u_i)} = \frac{\sum_{i=1}^{l} u_i \cdot max\left[\mu_{Bu}^{k}(u_i)\right]}{\sum_{i=1}^{l} max[\mu_{Bu}^{k}(u_i)]} \tag{4.4}$$

$$u^* = \frac{\int_{U_u} u \cdot \mu_{Bu}(u)du}{\int_{U_u} \mu_{Bu}(u)du} = \frac{\int_{U_u} u \cdot max\left[\mu_{Bu}^{k}(u)\right]du}{\int_{U_u} max[\mu_{Bu}^{k}(u)]du} \tag{4.5}$$

unde u* este valoarea singulară de ieșire, iar $\mu_{Bu}(u)$ este funcția de apartenență

Dificultatea acestei metode constă în determinarea funcției de apartenență a suprafeței totale rezultate, ceea ce poate îngreuna procesul de defuzzyficare.

Metoda centrului sumelor

Metoda centrului sumelor este asemănătoare centrului de greutate, diferența constă în faptul că se calculează centrul de greutate a fiecărei suprafețe componente a suprafeței totale, și apoi din aceste valori se calculează valoarea finală.

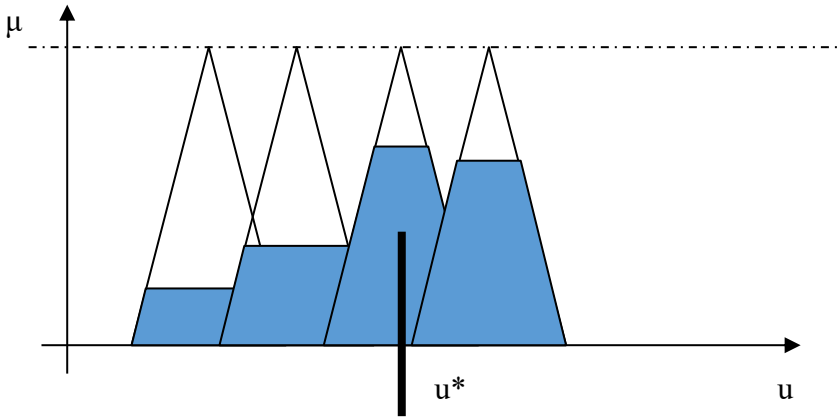



**Fig. 4.4.** Defuzzyficare – metoda centrului de greutate

Comparativ cu metoda centrului de greutate, în cazul acestei metode se integrează suma funcțiilor de apartenență inițiale. Astfel, din punctul de vedere a complexității funcției de apartenență, această metodă este mai simplă decât anterioara. Relațiile de calcul pentru funcții discrete și continue sunt (4.6) și (4.7). Fig. 4.5. prezintă grafic metoda centrului sumelor.

$$u^* = \frac{\sum_{i=1}^{l} u_i \sum_{k=1}^{n} \mu_{Bu}^{k}(u_i)}{\sum_{i=1}^{l} \sum_{k=1}^{n} \mu_{Bu}^{k}(u_i)} \tag{4.6}$$

$$u^* = \frac{\int_{U_u} u_i \sum_{k=1}^{n} \mu_{Bu}^{k}(u_i)}{\int_{U_u} \sum_{k=1}^{n} \mu_{Bu}^{k}(u_i)} \tag{4.7}$$

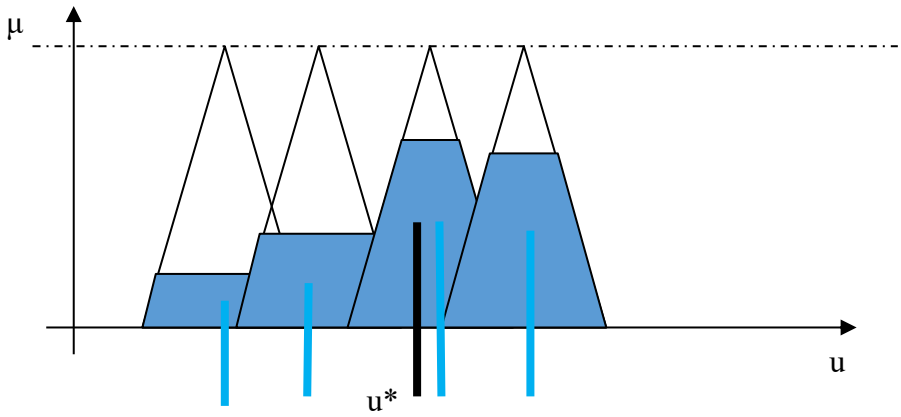



**Fig. 4.5.** Defuzzyficare – metoda centrului sumelor

Metoda bisectoarei

Această metodă determină valoare abscise care îndeplinește condiția – împarte suprafața fuzzy de ieșire în două suprafețe egale. Comparative cu metoda centrului de greutate, această metodă se focusează pe împărțirea echilibrată a funcțiilor de apartenență în două părți egale. Relația matematică care definește această metodă este (4.8). Comparativ cu metoda centrului de greutate care nu dă rezultate bune în cazul suprafețelor nesimetrice, această metodă este dedicată defuzzyficării acestui tip de suprafețe fuzzy. De asemenea, metoda presupune mai puține calcule decât celelalte două metode.

$$\int_{x_{min}}^{x} \mu_A(x) = \int_{x}^{x_{max}} \mu_A(x) \tag{4.8}$$

Metoda mediei poderate

Metoda mediei ponderate este simplu de utilizat și dedicată în special în situația în care se lucrează cu funcții discrete. Acesta calculează ieșirea singulară făcând o medie a valorilor funcțiilor fuzzy, în timp ce atribuie ponderi pe baza funcțiilor de apartenență a acestora. Relație de calcul pentru o funcție discretă este:

$$u^* = \frac{\sum_{i=1}^{l} u_i \cdot \mu_{Bu}(u_i)}{\sum_{i=1}^{l} \mu_{Bu}(u_i)} \tag{4.9}$$

unde $u_i$ este o valoare pe axa 0x, iar $\mu_{Bu}(u_i)$ este valoarea funcție de apartenență a acelei valori.

Metoda înălțimii

Aceasta este o metodă de defuzzyficare care se poate folosi cu rezultate bune în cazul suprafețelor simetrice. Ea este asemănătoare metodei precedente, diferența constă în faptul că se iau în calcul doar valorile maxime ale funcțiilor de apartenență, adică înălțimile. Relația de calcul este (4.10), cu $h_i$ fiind înălțimea corespunzătoare valorii $u_i$.

$$u^* = \frac{\sum_{i=1}^{l} u_i \cdot h_i}{\sum_{i=1}^{l} h_i} \tag{4.10}$$

Metoda centrului suprafeței celei mai mari

Această metodă presupune determinarea suprefeței celei mari mari care are funcția de apartenență continuă, pentru care se determină centrul de greutate, valoare care va fi folosită ca ieșire singulară.

$$u^* = \frac{\int_{U_{umax}} u \cdot \mu_{Bu}(u) du}{\int_{U_{umax}} \mu_{Bu}(u) du} \tag{4.11}$$

Metoda primului / ultimului dintre maxime

Din intervalul de încredere a suprafeței fuzzy de ieșire, se determină valoarea pentru care funcția de apartenență are prima / ultima valoare maximă.

$$u^* = \max\left(\mu_A(u)\right) \tag{4.12}$$

Grafic, metoda primului dintre maxime este explicată în figuta 4.6.

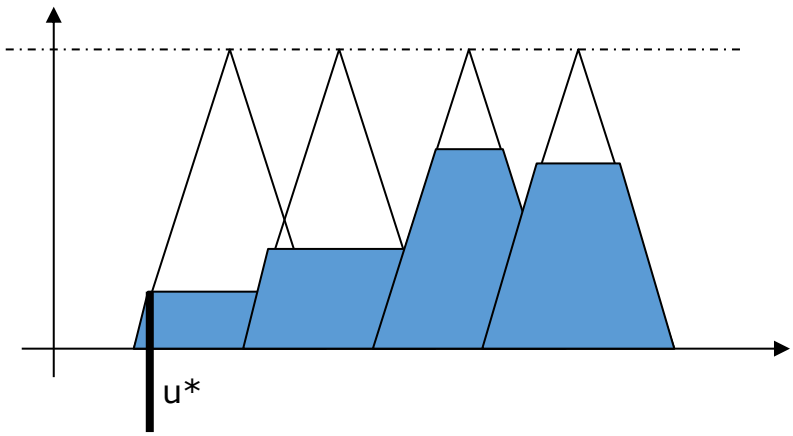



**Fig.4.6.** Defuzyficarea prin metoda primului dintre maxime

Metoda maximului din mijloc

Metoda maximului din mijloc constă în determinarea tuturor valorilor maxime, adică a înălțimilor, iar apoi se determină media lor.

O sinteză a punctelor forte și slabe a metodelor de defuzzyficare menționate anterior este realizată în tabelul 4.3.

**Tabelul 4.3.** Metode de defuzzyficare

| **Metodă defuzzyficare** | **Avantaje** | **Dezavanataje** |
|---|---|---|
| Centrului de greutate | -oferă o ieșire intuitivă și echilibrată<br>-rezultat robust (folosește toată funcția de apartenență)<br>- ieșire clară, reprezentativă pentru comportamentul general al ieșirii<br>-potrivit pentru funcțiile de apartenență continue. | -pentru suprafețe complexe calculele se complică, ducând la calcule extensive<br>-nu funcționează bine pentru funcțiile discrete<br>-sensibilă la formă – nu dă rezultate bune pentru suprafețe nesimetrice |
| Centrului sumelor | -funcționează bine pentru sistemele bazate pe reguli: calculează media ponderată a seturilor fuzzy individuale, care poate fi benefică atunci când se combină mai multe reguli<br>-mai simplu din punct de vedere computațional decât centrul de greutate<br>-gestionare mai bună a numerelor fuzzy multiple – nici o regulă nu domină, luând în considerare toate numere fuzzy separat | -regiunile care se suprapun sunt numărate de mai multe ori -> poate distorsiona rezultatul în unele cazuri, mai ales când regulile se suprapun în mod semnificativ<br>-valoarea de ieșire singulară rezultată poate fi denaturată dacă numerele fuzzy de ieșire sunt neregulate<br>-mai puțin precisă decât centrul de greutate |
| Bisectoarei | -intuitiv: împarte aria de sub curbă în două părți egale, care pot fi ușor de înțeles și aplicat în luarea deciziilor | -mai puțin precis decât centru de greutate |

| | | |
|---|---|---|
| | -bună pentru suprafețe fuzzy asimetrice, se concentrază pe găsirea unei abscise nu pe centru de greutate<br>-mai simplă din punct de vedere computațional decât centru de greutate | -poate să nu dea un rezultat semnificativ pentru luarea deciziilor în sistem<br>-poate fi dificil de calculat analitic pentru funcții de apartenență continue. |
| Înălțimii | -simplă și intuitivă<br>-eficientă pentru suprafețe fuzzy triunghiulare și trapezoidale<br>-rapidă în calcul - se concentrează pe cea mai mare valoare de membru, evită integrarea complexă | -se ia în considerare doar înălțimea și nu lățimea sau forma generală a numărului fuzzy<br>-dacă funcția de apartenență are mai multe înălțimi, metoda poate alege prima sau cea mai înaltă, care poate să nu fie reprezentativ pentru suprafața fuzzy totală<br>-nu este potrivită pentru forme complexe - este posibil să nu funcționeze bine cu funcțiile de apartenență neregulate sau asimetrice |
| Mediei ponderată | -simplă și rapidă - este ușor de calculat și nu necesită integrări complexe<br>-funcționează bine cu numere fuzzy discrete<br>-eficientă pentru probleme mici - poate fi folosită pentru aproximări rapide în aplicații simple | -folosește doar medii ponderate și nu ține cont de forma completă a funcției de apartenență<br>-nu este ideal pentru numere fuzzy continue<br>-valorile extreme din funcția de apartenență pot influența în mod disproporționat rezultatul |
| Maximului suprafeței maxime | -se concentrează pe valori mai mari din numărului fuzzy, ceea | -poate ignora valori mai mici, dar potențial |

| | | |
|---|---|---|
| | ce poate fi benefic pentru aplicații specifice de control<br>-simplu de calculat<br>-funcționează bine pentru sistemele de decizie care prioritizează rezultate mai mari | semnificative din suprafața fuzzy<br>-poate duce la valori extreme<br>-nu ține cont de distribuția generală a numărului fuzzy, care ar putea fi importantă în unele aplicații |
| Primului maxim | -simplă și rapidă<br>-asigură că rezultatul se bazează pe primul vârf al setului fuzzy, care poate fi preferat în anumite aplicații<br>-ușor de implementat: este eficientă din punct de vedere computațional și nu necesită operațiuni complexe | -ignoră forma funcției: poate fi insensibilă la distribuția valorilor funcției de apartenență în afara primului maxim<br>-poate fi înșelător dacă funcția de apartenență nu este definită clar |
| Maximului din mijloc | -rezultat echilibrat – determină media tuturor punctelor maxime, ceea ce ajută la evitarea extremelor și oferă un rezultat stabil și echilibrat<br>-mai puțin sensibilă la valori aberante<br>-bună pentru numerele fuzzy simetrice - deosebit de eficientă atunci când există mai multe valori maxime | -mai puțin precisă pentru formele asimetrice<br>-în sistemele cu multe puncte maxime, poate fi mai greu din punct de vedere computațional decât alte metode<br>-nu este ideală pentru seturi discrete - poate să nu funcționeze bine atunci când numărul fuzzy nu este continuu |

***Funcționarea unui controler fuzzy*** se realizează conform schemei de principiu din fig. 4.7. Astfel, în prima etapă datele primite de la exterior (notate cu x) ca date de intrare vor ajunge în blocul de fuzzyficare. În acest bloc sunt determinate valorile lingvistice și valorile funcțiilor de apartenență corespunzătoare numerelor fuzzy de intrare. Aceste date sunt transmise mai departe blocului de reguli și algoritmului de rezolvare a inferențelor. Din blocul

de reguli sunt extrase acele reguli care au în partea de premisă (condiție) valorile lingvistice găsite în etapa ulterioară. Aceste reguli sunt folosite pentru a rezolva inferențele prin aplicarea metodei alese. Această etapă va duce la obținerea unei suprafețe fuzzy la ieșire, care ajunge în blocul de defuzzyficare, unde prin aplicarea metodei de defuzzyficare implementate la început se obține valoarea singulară de la ieșire (notată cu y).

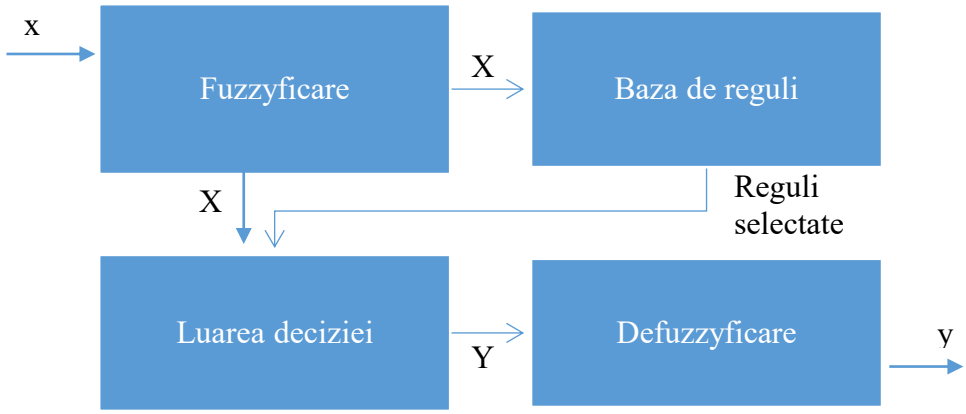



**Fig. 4.7.** Schema funcțională a unui regulator fuzzy. x – mărime fizică singulară, X – număr fuzzy, Y – mărime de ieșire fuzzy, y – mărime singulară

În concluzie, în funcționarea regulatorului fuzzy sunt realizați 4 pași:

1. Folosind datele de intrare, se determină valorile lingvistice necesare și se calculează valorile funcțiilor de apartenență atașate acestor valori lingvistice (numere fuzzy), mai exact se realizează o prelucrare preliminară a datelor de intrare și tratează aceste mărimi prin variabile lingvistice atașate.
2. Se extrag din blocul de reguli, acele reguli care corespund datelor de valorilor lingvistice determinate l apasul 1.
3. Se determină suprafața fuzzy care corespunde mărimii de ieșire.
4. Se aplică metoda de defuzzyficare care caracterizează cel mai fidel procesul reglat, astfel obținându-se valoarea singulară de ieșire.

Modul în care funcționează un controler fuzzy este descris în Anexa 1. În contiuare sunt prezentate două exemple numerice care arată metoda de luare a

decizie în cazul unei inferențe cu o singură regulă, respectiv a unei inferențe cu mai multe reguli.

***Exemplu numeric (inferența cu o singură regulă)***

Se consideră controlul unui aparat de aer condiționat cu patru mărimi de intrare și valorile funcțiilor de apartenență

x1 – temperatura = 25°C, funcția de apartenență 0,3;

x2 – viteza = 50 km/h, funcția de apartenență 0,6;

x3 – presiune = 30 kPa, funcția de apartenență 0,4;

x4 – umiditate = 70 %, funcția de apartenență 0,8.

Regula este u = x1 SAU (x2 ȘI x3) SAU x4.

La final u = 0,8. (u = max (0,3, min (0,6, 0,4), 0,8)).

***Exemplu numeric (inferența cu mai multe reguli)***

Se consideră un controler fuzzy care are trei date de intrare (i1, i2, i3) și o ieșire, y. În timpul funcționării, blocul de fuzzyficare oferă următoarele valori lingvistice și valori ale funcțiilor de apartenență (funcții triunghiulare)

a1, valori lingvistice A, B, funcția de apartenență A - 0,4, iar B – 0,7;

a2, valori lingvistice B, C, funcția de apartenență B - 0,9, iar C – 0,2;

a3, valori lingvistice A, B, funcția de apartenență A - 0,5, iar B – 0,6.

Prin folosirea acestor informații, se vor selecta următoarele reguli:

1. Dacă a1=A(0,4) și a2=B(0,9) și a3=A(0,5), Atunci y=A, SAU
2. Dacă a1=A(0,4) și a2=B(0,9) și a3=B(0,6), Atunci y=A, SAU
3. Dacă a1=A(0,4) și a2=C(0,2) și a3=A(0,5), Atunci y=B, SAU
4. Dacă a1=A(0,4) și a2=C(0,2) și a3=B(0,6), Atunci y=B, SAU
5. Dacă a1=B(0,7) și a2=B(0,9) și a3=A(0,5), Atunci y=B, SAU
6. Dacă a1=B(0,7) și a2=B(0,9) și a3=B(0,6), Atunci y=C, SAU
7. Dacă a1=B(0,7) și a2=C(0,2) și a3=A(0,5), Atunci y=B, SAU
8. Dacă a1=B(0,7) și a2=C(0,2) și a3=B(0,6), Atunci y=C, SAU

Această listă de reguli și valorile funcțiilor de apartenență devin datele de intrare ale algoritmului de rezolvare a inferențelor. Rezultatul metodei de rezolvare a inferențelor reprezintă suprafața fuzzy de ieșire, care ajunge la blocul de defuzzyficare. În continuare sunt explicate metodele de rezolvare a inferențelor, pentru fiecare regulă.

Metoda Max-Min

Regula 1 – min(0,4; 0,9; 0,5)->0,4 -> min(0,4; y=A) - >număr trapezoidal, înălțimea 0,4.

R2 - min(0,4; 0,9; 0,6)->0,4 -> min(0,4; y=A) - >număr trapezoidal, h=0,4.

R3 - min(0,4; 0,2; 0,5)->0,2 -> min(0,2; y=B) - >număr trapezoidal, h=0,2.

R4 - min(0,4; 0,2; 0,6)->0,2 -> min(0,2; y=B) - >număr trapezoidal, h=0,2.

R5 - min(0,7; 0,9; 0,5)->0,5 -> min(0,5; y=B) - >număr trapezoidal, h=0,5.

R6 - min(0,7; 0,9; 0,6)->0,6 -> min(0,6; y=C) - >număr trapezoidal, h=0,6.

R7 - min(0,7; 0,2; 0,5)->0,2 -> min(0,2; y=B) - >număr trapezoidal, h=0,2.

R8 - min(0,7; 0,2; 0,6)->0,2 -> min(0,2; y=C) - >număr trapezoidal, h=0,2.

Se va realiza maximul dintre valorile funcțiilor de apartenență pe intervalul de încredere a variabilei de ieșire y. La final, suprafața de ieșire Y va cuprinde trei numere fuzzy trapezoidale, cu caracteristicile y=A(0,4), y=B(0,5) și y=C(0,6).

Metoda Max-Prod

R1 – min(0,4; 0,9; 0,5)->0,4 -> prod(0,4; y=A) - >număr triunghiular, h=0,4.

R2 - min(0,4; 0,9; 0,6)->0,4 -> prod(0,4; y=A) - >număr triunghiular, h=0,4.

R3 - min(0,4; 0,2; 0,5)->0,2 -> prod(0,2; y=B) - >număr triunghiular, h=0,2.

R4 - min(0,4; 0,2; 0,6)->0,2 -> prod(0,2; y=B) - >număr triunghiular, h=0,2.

R5 - min(0,7; 0,9; 0,5)->0,5 -> prod(0,5; y=B) - >număr triunghiular, h=0,5.

R6 - min(0,7; 0,9; 0,6)->0,6 -> prod(0,6; y=C) - >număr triunghiular, h=0,6.

R7 - min(0,7; 0,2; 0,5)->0,2 -> prod(0,2; y=B) - >număr triunghiular, h=0,2.

R8 - min(0,7; 0,2; 0,6)->0,2 -> prod(0,2; y=C) - >număr triunghiular, h=0,2.

Se va realiza maximul dintre valorile funcțiilor de apartenență pe intervalul de încredere a variabilei de ieșire y. La final, suprafața de ieșire Y va cuprinde trei numere fuzzy triunghiulare, cu caracteristicile y=A(0,4), y=B(0,5) și y=C(0,6).

Metoda Sum-Prod

R1 – prod(0,4; 0,9; 0,5)->0,18 -> prod(0,18; y=A) - >număr triunghiular, h=0,18.

R2 - prod(0,4; 0,9; 0,6)->0,21 -> prod(0,21; y=A) - >număr triunghiular, h=0,21.

R3 - prod(0,4; 0,2; 0,5)->0,04 -> prod(0,04; y=B) - >număr triunghiular, h=0,04.

R4 - prod(0,4; 0,2; 0,6)->0,064 -> prod(0,064; y=B) - >număr triunghiular, h=0,064.

R5 - prod(0,7; 0,9; 0,5)->0,31 -> prod(0,31; y=B) ->număr triunghiular, h=0,31.

R6 - prod(0,7; 0,9; 0,6)->0,37 -> prod(0,37; y=C) ->număr triunghiular, h=0,37.

R7 - prod(0,7; 0,2; 0,5)->0,07 -> prod(0,07; y=B) ->număr triunghiular, h=0,07.
R8 - prod(0,7; 0,2; 0,6)->0,084 -> prod(0,084; y=C) ->număr triunghiular, h=0,084.

Se va realiza suma dintre valorile funcțiilor de apartenență pe intervalul de încredere a variabilei de ieșire y. La final, suprafața de ieșire Y va cuprinde trei numere fuzzy triunghiulare, cu caracteristicile y=A(0,39), y=B(0,484) și y=C(0,454).

## 4.3. Sisteme fuzzy convenționale

Sistemele fuzzy de control au reprezentat subiectul capitolului precedent, în continuare sunt prezentate caracteristice altor tipuri de sistemelor fuzzy convenționale menționate în capitolul 4.1: sisteme de prognoză, sisteme de optimizare, sisteme decizionale și de identificare/recunoaștere a modelelor (pattern recognition).

### 4.3.1. Sisteme fuzzy decizionale

Sistemele fuzzy decizionale sunt folosite în diagnoza defectelor (identificarea și clasificarea defectelor în rețelele de transport și distribuție), dispecerizarea consumului (identificarea momentului, a duratei și a consumatorului alimentat în eventimentul unei congesti), operații de comutare (ajută la reconfigurarea rețelei după o avarie) și distribuirea optimă a puterii (alegerea combinației optime de funcționare a generatoarelor ăn funcție de diverși factori, precum costul).

Un sistem fuzzy decizional are aceeași structură (fig. 4.1) și funcționare (fig. 4.7) ca un sistem de control, diferența constă în modul în care este utilizat sistemul și rolul lui. Asftel, prima etapă este determinarea variabilelor care influențează decizia, și a ieșierii, urmată de fuzzyficarea mărimilor de intrare și ieșire. La fel, ca la sistemele fuzzy de control, se construiește baza de reguli, se implementează metoda de rezolvare a inferențelor, iar ultima etapă este defuzzyficarea. Toate etapele enumerate anterior se realizează conform informațiilor menționate la capitolul 4.2.

### 4.3.2 Sisteme fuzzy de optimizare

Sistemele fuzzy de optimizare în prezent sunt utilizate cu succes în următoarele probleme din energetică: integrarea energiei regenerabile (optimizarea utilizării surselor regenerabile și convenționale), circulația de putere

optimizată (optimizarea circulației de putere în vederea scăderii costului și menținerea stabilității), evaluarea fiabilității sistemelor electroenergetice etc.

Un sistem fuzzy de optimizare conține elementele unui sistem fuzzy de control (vezi fig. 4.2), la care se adaugă funcția obiectiv, care se minimizează sau maximizează, plus constrângeriile / restricțiile necesare. Funcționarea sistemului fuzzy de optimizare se desfășoară la fel ca a unui sistem de control, adică fuzzyficarea, rezolvarea inferențelor și defuzzyficarea, cu diferența că rezolvarea inferențelor se va face doar dacă sunt respectate restricțiile și după determinarea funcției obiectiv.

Funcția obiectiv și restricțiile sunt caracteristice problemei de rezolvat. De exemplu, pentru problema optimizării circulației de putere, funcția obiectiv poate fi reducerea pierderilor de putere în rețele, iar constrângeriile: limitarea variației tensiunii la 90-110% $U_n$ (tensiunea nominală a rețelei) și limitarea curentului sub limita impusă prin standard.

### 4.3.3. Sisteme fuzzy de recunoaștere a modelelor

Sistemele fuzzy de recunoaștere a modelelor se folosesc pentru detecția și diagnoza avariilor, clasificarea consumatoriilor, monitorizarea calității energiei electrice, localizarea defectelor și monitorizarea funcționării echipamentelor.

Un sistem fuzzy de recunoaștere a modelelor are aceeași structură și funcționalitate ca a unei sistem fuzzy decizional; diferența constă în rolul și aplicabilitatea sistemului fuzzy. Deci, se determină variabilele care influențează modelul, iar ieșirea indică tipul modelului. Următorul pas este fuzzyficarea variabilelor, definirea regulilor, a metodei de rezolvarea a inferențelor, și a metodei de defuzzyficare. La primirea unor date de intrare singulare, se determină valorile lingvistice și valorile funcțiilor de apartenență, precum și lista cu reguli care se aplică. Apoi se rezolvă inferențele, și de realizează defuzzyficarea.

### 4.3.4. Sisteme fuzzy de prognoză

Sistemele fuzzy de prognoză sunt folosite în aplicații precum:

- Prognoza consumului de energiei electrică;
- Prognoza resurselor regenerabile;
- Prognoza producției de energie electrică;
- Predicția problemelor de calitate în sistemelor eletroenergetice.

Un sistem fuzzy de prognoză are aceeași structură și funcționare ca un sistem fuzzy decizional, doar că se aplică în cazul problemelor de prognoză / predicție.

La fel ca în cazul sistemului fuzzy decizional, sistemul cuprinde blocul de fuzzyficare, baza de reguli, algoritmul de rezolvare a inferențelor și blocul de defuzzyficare. Funcționarea acestui tip de sistem este la fel ca celelalte tipuri (control, decizional, de recunoaștere a modelelor).

În concluzie, legat de aceste sisteme se poate spune că, în afară de sistemele de optimizare, celelalte patru tipuri de sisteme fuzzy (decizie, prognoză, recunoaștere a modelelor, control) au aceeași structură și mod de funcționare.

Sistemele fuzzy bazate pe inferențe cu mai multe reguli prezentate până acum au fost introduse de Ebrahim Mamdani. Ele se caracterizează prin flexibilitate, dar și prin faptul că dau rezultate intuitive, iar funcțiile de apartenență pot fi neliniare. Însă, punctele slabe precum numărul mare de reguli și numărul mare de calcule necesare, în unele cazuri descurajează specialiști în utilizarea lor. Alternative pentru sistemele Mamdani sunt sisteme fuzzy Takagi-Sugeno și sistemele fuzzy de tipul 2.

### 4.3.5. Sisteme fuzzy Sugeno

Sistemele fuzy Takagi-Sugeno-Kang, sau simplu sisteme Sugeno se diferențiază de sistemele de tip Mamdani prin faptul că ieșirea nu este reprezentată de o suptafață fuzzy, ci printr-o funcție matematică. Această caracteristică dauc un plus sistemelor Sugeno, prin faptul că le fac mai rapide și dedicate în special pentru control și prognoză.

Structura unui sistem Sugeno conține blocul de fuzzyficare, baza de reguli, metoda de rezolvare a inferențelor și blocul de defuzzyficare.

Procesul de fuzzyficare este la fel ca în cazul sistemelor Mamdani, adică se determină pentru datele de intrare variabilele lingvistice, valorile lor lingvistice și numere fuzzy asociate.

Baza de reguli conține toate regulile care fac asocierea dintre datele de intrare și ieșire. Forma unei reguli într-un sistem Sugeno este

Dacă x1=A1 și x2=A2, Atunci y=f(x1, x2)

Unde x1 și x2 sunt variabilele fuzzy de la intrare, A1 și A2 sunt valorile lingvistice și numere fuzzy asociate, iar f(x1, x2) este funcția de ieșire. Astfel se poate observa că ieșierea după realizarea inferenței este o valoare singulară, ci nu un număr fuzzy.

Funcție de ieșire a sistemului fuzzy Sugeno poate să ia următoarele forme:

- Model Sugeno de ordinul zero – ieșirea este o valoare constantă

  y = c (constantă);

- Model Sugeno de ordinul unu – ieșirea este o funcție liniară

  y = a*x1 + b*x2 + c;

- Model Sugeno de ordinul înalt – funcția de ieșire este o funcția polimonială de ordin mare.

Defuzzyficarea ieșirii unui sistem Sugeno se realizează folosind media ponderată

$$y = \frac{\sum w_i f_i}{\sum w_i} \tag{4.12}$$

Unde $w_i$ este ponderea regulii i (determinată din funcțiile de apartenență, și apată cât de mult o regulă influențează valoarea de ieșire), iar $f_i$ – valoarea de ieșire a regulii i. Ponderea regulii menționate anterior se calculează astfel:

$$w_i = \mu_{A1}(x1) \cdot \mu_{A2}(x2). \tag{4.13}$$

### 4.3.6. Sisteme fuzzy de tipul 2

Sistemele fuzzy de ordinul 2 sunt o extensie a sistemelor fuzzy de tip 1 (cele prezentate până acum). Aceste sisteme fuzzy au capabilitatea de a lucra cu date influențate de zgomot accentuat, sau incertitudine ridicată, sau variază în timp.

Un sistem fuzzy de tipul 2 este similar celui de tipul 1, singura diferență constă în firma și tipul funcțiile apartenență. În cazul sistemelor fuzzy de gradul 1, valorile funcțiilor sunt valori singulare, care iau valoarea între 0 și 1. În cazul sistemelor fuzzy de gradul 2, valorile funcțiilor de apartenență sunt numere fuzzy.

Structura unui sistem fuzzy de gradul 2 cuprinde următoarele componente:

- Blocul de defuzzyficare – transformă mărimile cu care lucrează sistemul în numere fuzzy de gradul 2 (valorile funcțiilor de apartenență sunt numere fuzzy).
- Baza de reguli – conține toate regulile care asociază datele de intrare și ieșirea. Regulile se scriu la fel ca în cazul sistemelor de tipul 1.
- Metoda de rezolvare a inferențelor – se folosesc aceleași metode ca în cazul sistemelor de tipul 1, doar că se lucrează nu cu numere singulare, și cu numere fuzzy. De exemplu, la ieșire nu va fi minimul dintre un număr fuzzy și un scalar, și minimum dintre două numere fuzzy, unul de tipul 1 și unul de tipul 2.
- Blocul de reducere – transformă numerele fuzzy de tipul 2 de la ieșire în numere fuzzy de tipul 1.

- Blocul de defuzzyficare – folosește aceleași metode ca în cazul sistemelor de tipul 1, deci transformă un număr fuzzy într-o valoare singulară de ieșire.

Numerele fuzzy de tipul 2 sunt definite prin funcții de apartenență care au trei caracteristici: funcția de apartenență superioară, funcția de apartenență inferioară și amprenta incertitudinii (distanța dintre funcția superioară și cea inferioară). Reprezentarea grafică a unui număr fuzzy de tip 2 este ilustrată în fig. 4.8.

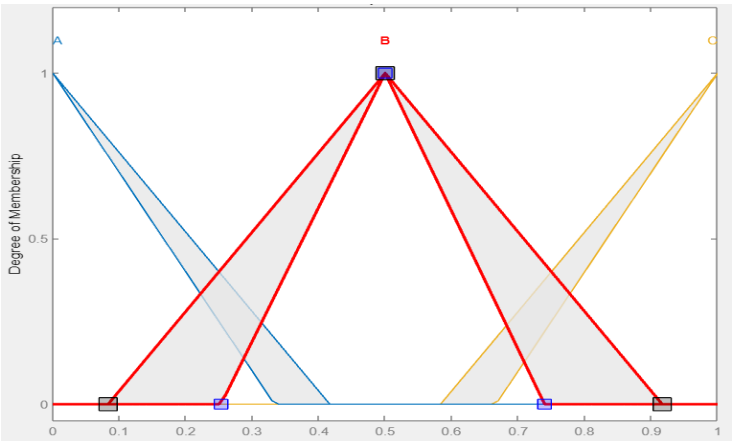



**Fig. 4.8.** Reprezentarea grafică a unui număr fuzzy de tipul 2 dezvoltat cu MatLab

Blocul de reducere are implementată o metodă de reducere. Două dintre cele mai populare metode de reducere a unei suprafețe fuzzy de tip 2 în suprafață fuzzy de tip 1 sunt: reducere de tipul centrul de greutate (algoritmul Karnik-Mendel) și metoda înălțimii sau a înălțimii modificate.

Avantajele acestui tip de sisteme fuzzy constau în capacitatea de a implementa mai fidel mărimi afectate de incertitudini și zgomot, performanță ridicată în cazul sistemelor complexe și luarea deciziei este mai robustă. Dezavantajele sistemelor de tip 2 sunt timpul mai mare de calcul și dificulatatea în implementare.

### 4.3.7. Sisteme fuzzy Matsumoto

Sistemele fuzzy Matsumoto , introduse de Toshio Matsumoto, sunt un tip special de sisteme fuzzy care diferă de sistemele Mamdani și Sugeno. Aceste sisteme se concentrează pe creșterea eficienței a proceselor de luare a deciziilor

și de control prin îmbunătățirea modului în care incertitudinea și imprecizia sunt gestionate în aplicațiile din lumea reală.

Modificările lui Matsumoto asupra sistemelor fuzzy integrează elemente de control adaptiv, auto-învățare și logica fuzzy ierarhică. Astfel, spre deosebire de sistemele fuzzy convenționale care se bazează pe funcții de apartenență predefinite și bază de reguli, modelul Matsumoto introduce mecanisme pentru auto-ajustare și învățare. Caracteristicile funcționale de bază ale unui sistem fuzzy Matsumoto sunt:

- Subliniază adaptarea dinamică: sistemul își poate modifica regulile fuzzy și funcțiile de apartenență pe baza datelor primite.
- Utilizează o structură ierarhică fuzzy: în loc de o bază de reguli plată, folosește mai multe straturi de procesare fuzzy.
- Combină logica fuzzy cu optimizarea: introduce mecanisme de îmbunătățire a performanței în timp.

Un sistem fuzzy Matsumoto cuprinde următoarele componente: blocul de fuzzyficare, baza de reguli ierarhice, metoda de rezolvare a regulilor ierarhice, algoritmul de învățare, algoritmul decizional optimizat, și blocul de defuzzyficare.

Sistemele fuzzy Matsumoto sunt mai puțin cunoscute decât sistemele Mamdani și Sugeno, dar aduce avantaje importante precum: adaptabilitatea la problema de rezolvat și luarea deciziei pe mai multe niveluri.

### 4.3.8. Sisteme fuzzy pentru suportul deciziei

În energetică, sistemele fuzzy pentru suportul deciziei sunt folosite ca asistenți în procesele decizionale și de management a operatorilor, a inginerilor și a specialiștilor din domeniu, în situații caracterizate de incertitudine și dinamism. Principalele aplicații sunt în managementul energetic al rețelelor electrice inteligente (smart grids) și evaluarea securității sistemelor electroenergetice. Utilizarea acestor tipuri de sisteme fuzzy este încurajată de avantajele aduse de acestea, și anume: lucrează cu incertitudini, oferă suport în timp real, posibilitatea interconectării cu alte tipuri de sisteme, precum bazele de date de dimensiuni mari, și crește stabilitatea și funcționalitatea sistemului.

Un sistem fuzzy pentru suportul deciziei cuprinde cinci componente definitorii: blocul de fuzzyficare, blocul de defuzzyficare, baza de reguli, algoritmul de rezolvare a inferențelor și interfața cu utilizatorul. În consecință, față de majoritatea sistemelor fuzzy, acest tip de sisteme fuzzy au un element în

plus, și anume interfața cu uitilizatorul, prin care sistemul comunică cu specialistul și în ajută în luarea deciziilor.

### 4.3.9. Sisteme fuzzy de grupare (clustering)

Sistemele fuzzy de grupare intră în categoria metodelor de grupare soft, care dau posibilitatea obiectelor de a face parte din mai multe grupuri. Rezultă că, relația unui obiect cu un grup este caracterizată prin funcții de apartenenență, care sunt reprezentate prin numere fuzzy. Condiția de bază în gruparea fuzzy este ca suma valorilor funcțiilor de apartenență a unui obiect să fie egală cu 1.

Un sistem fuzzy de grupare este compus din următoarele componente:

- Mulțimea obiectelor (punctelor) care urmează a fi grupate;
- Funcțiile de apartenență ale tuturor punctelor care aparțin diferitelor grupuri;
- Centrele grupurilor – punctele reprezentative a fiecărui grup;
- Metoda de grupare - metoda de evaluare a grupării prin măsurarea distanței a punctelor față de diferite grupuri;
- Criteriu de optimizare – funcția de optimizare dedicată micșorării distanței dintre puncte și diferite grupuri.

Un sistem fuzzy de grupare are implementat un algoritm de grupare, prin care se realizează gruparea supravegheată sau nesupraveghetaă a obiectelor. Există mulți algoritmi de grupare, dar cel mai popular este algoritmul fuzzy C-Means, care va fi descris în continuare.

Algoritmul fuzzy C-Means are ca principală acțiune minimizarea sumei distanțelor dintre punctele mulțimii inițiale și grupurile definite. Două caracteristici a algoritmului C-Means este că numărul de grupuri este specificat, și este permisă suprapunerea funcțiile de apartenență a punctelor. Etapele în funcționarea acestui algoritm sunt următoarele: (1) inițializarea cu valori arbitrare a centrelor grupurilor definite, (2) calcularea valorilor funcțiilor de apartenență  și a gradului de apartenență a punctelor folosind relația (4.14), (3) recalcularea punctelor de centru a grupurilor folosind media valorilor funcțiilor de apartenență conform relația (4.15), și (4) repetarea etapelor 2 și 3 până când punctele de centru a grupurilor au convers spre o valoare fixă.

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{ik}}\right)^{\frac{2}{m-1}}} \tag{4.14}$$

Unde $d_{ij}$ este distanța dintre punctul $x_i$ și grupul $C_j$, este parametru fuzzy, cu proprietatea că este supraunitar.

$$C_j = \frac{\sum_{i=1}^{N} \mu_{ij}^m \cdot x_i}{\sum_{i=1}^{N} \mu_{ij}^m} \tag{4.15}$$

Din relațiile anterioare se observă că, cu cât un punct este mai aproape de un grup, cu atât gradul de apartenență este mai mare, în caz contrar va fi mai mic.

## 4.4. Realizarea sistemelor fuzzy

Sistemele fuzzy pot fi realizate în varianta soft (software-based fuzzy systems) și în varianta hard (hardware-based fuzzy systems).

4.4.1 Sisteme fuzzy bazate pe applicații informatice

Sisteme fuzzy software-based sunt implementate folosind medii, unelte informatice și limbaje de programare dedicate pentru construcția, simularea și testarea acestora. Cele mai populare instrumente informatice sunt: Fuzzy Logic Designer (MATLAB), Phyton (scikit-Fuzzy, py-Fuzzy), LABVIEW Fuzzy Logic Toolkit, Fuzzy TECH, Juzzy (Java Fuzzy Logic Library), Octave Fuzzy Logic (Octave-Fuzzy), și alte medii de programare încorporate (C, C++, Phyton).

Această abordare în implementarea sistemelor fuzzy oferă multe avantaje precum:

- Ușor de implementat și testat;
- Ideal pentru simulări și realizarea de prototipuri;
- Poate ușor integra și alte tehnici de IA.

care fac din aceasta cea mai populară abordare în rândul specialiștilor.

Fuzzy Logic Designer, FLD de la MATLAB este o unealtă dedicată realizării de sisteme fuzzy bazate pe software. Aceasta face parte din Fuzzy Logic Toolbox, care oferă o interfață grafică interactivă și funcții dedicate pentru dezvoltarea și simularea de sisteme fuzzy de interfață (aceste fișiere în MATLAB au extensia FIS).

FDL suportă dezvoltarea de sisteme fuzzy de două tipuri principale, și anume sisteme Mamdani și sisteme Sugeno. În plus față de acestea, FDL are capacitatea de a lucra cu sisteme hibride de tipul ANFIS și sisteme fuzzy de tipul 2. Sistemele de tipul Mamdani sunt folosite în special în control, iar cele de tipul Sugeno în probleme de optimizare și control adaptiv.

Punctele forte ale FDL sunt:

- Interfață grafică prietenoasă – nu sunt necesare cunoștințe avansate de logica fuzzy și programare pentru a o folosi;
- Simulare și vizualizare – dă posibilitatea utilizatorilor să testeze sistemele fuzzy prin introducere datelor de intrare și vizualizarea rezolvării.
- Posibilitatea personalizării – utilizatorii pot să își definească funcțiile de apartenență, regulile și metodele de rezolvare a inferențelor într-un mod simplu și accesibil;
- Integrarea cu Simulink – sistemele fuzzy dezvoltate cu FDL pot fi integrate în modele Simulink, și apoi folosite în aplicații și simulări în timp real;
- Compatibilitate cu IA și optimizări – suportă conectarea cu rețele neuronale artificiale și algoritmi de optimizare;
- Coduri MATLAB – utilizatorii pot să automatizeze dezvoltarea sistemelor fuzzy folosind linii de comandă în MATLAB.

Utilizarea FDL pentru dezvoltarea sistemelor fuzzy oferă multe avantaje, dar are și unele limitări precum costul (MATLAB este un software comercial), probleme de performanță pentru sistemele de dimensiuni mari (timpul de lucru a sistemelor fuzzy complexe poate crește mult), nu este ideal pentru sisteme integrate (sunt necesari pași suplimentari care pot pune în dificultate utilizatori) și limitarea la sisteme fuzzy de tipul 1 (chiar dacă se pot dezvolta sisteme fuzzy de tipul 2, însă acestea nu sunt suportate natural, ci pleacă de la sisteme de tipul 1).

Fuzzy Logic Toolkit, FLT de la LabVIEW cuprinde mai multe unelte software dedicate pentru modelarea, simularea și utilizarea sistemelor fuzzy, prin implicarea mediului grafic de programare LabVIEW. Astfel, FLT este potrivit pentru controlul în timp real și aplicații de automatizare în sisteme industriale.

Acest ansamblu de unelte LabVIEW suportă cele două categorii mari de système fuzzy, și anume système fuzzy Mamdani și Sugeno. De asemenea, la fel ca FDL, FLT dă posibilitatea utilizatorilor să construiască sisteme hibride compuse din sisteme fuzzy și controlere PID, rețele neuronale și algoritmi de optimizare, dar și implementarea hardware a sistemelor fuzzy pe FPGA, microprocesoare dedicate și aparatură de automatizare industrială.

Avantajele uneltelor FLT sunt următoarele:

- Interfață grafică de programare – se utilizează metoda ”drag – and - drop” pentru a dezvolta sistemele fuzzy, ceea ce ușurează procesul de obținere a sistemelor fuzzy;
- Control în timp real și aplicații industriale;
- Integrarea cu senzori și actuatori – dă posibilitatea interacțiunii directe cu semnale citite direct în timp real, pentru testarea HIL (hardware-in-the-loop);
- Personalizare flexibilă – utilizatorii pot să definească funcțiile de apartenență, regulile și metodele de inferențe;
- Mediu de simulare tip Simulink – suportă testarea în timp real a sistemelor fuzzy prin folosirea LabVIEW Real-Time & FPGA.

În dezvoltarea sistemelor fuzzy trebuie avut în vedere punctele slabe ale FLT, și anume: limitarea la sisteme fuzzy de tipul 2, necesitatea cunoașterii programării în mediul grafic LabVIEW, cost suplimentar deoarece LabVIEW este un software comercial, și mai puțin flexibil când se implementează alte tehnici de IA și învățarea automată.

Fuzzy TECH este un instrument software specializat pentru dezvoltarea, analiza și implementarea sistemelor fuzzy. Acesta este utilizat extensiv în industrie, sisteme de automatizare și aplicații pentru suportul decizional. Fuzzy TECH suportă dezvoltarea de sisteme fuzzy de tipul Mamdani, Sugeno, Neuro-Fuzzy și hibride. În plus față de acestea, acest software oferă posibilitatea ca utilizatorii să dezvolte sisteme fuzzy ierarhice și integrate cu sisteme hardware de tipul microcontrolere, PLC-uri și sisteme de automatizare industriale.

Punctele forte a fuzzy TECH sunt:

- Mediu de programare dedicat pentru dezvoltarea sistemelor fuzzy – este un instrument software optimizat pentru construirea de aplicații ale logicii fuzzy;
- Editarea grafică a regulilor – are o interfață cu modele drag-and-drop atât pentru reguli cât și pentru funcții de apartenență;
- Simulare și analiză în timp real – oferă instrumente vizuale puternice pentru testarea și optimizarea sistemelor fuzzy;
- Dezvoltarea de sisteme integrate – poate genera cod sursă de tip C, VHDL, și compatibil cu PLC pentru aplicații de control în timp real;
- Integrarea cu automatizarea industrială – lucrează cu SCADA, PLC-uri și microprocesoare pentru aplicații reale;

- Suportă abordări hibride – dă posibilitatea utilizatorilor să combine logica fuzzy cu rețelele neuronale artificiale și învățarea automată.

La fel ca celelalte instrumente software prezentate anterior, fuzzy TECH este un produs comercial, ceea ce îl face greu de accesat pentru mulți utilizatori. De asemenea, acest software este limitat doar pentru logica fuzzy, și în primul rând dedicat pentru sisteme fuzzy de tip 1. Un alt punct slab al acestui instrument este faptul că nu este folosit foarte mult în cercetare.

Octave Fuzzy Logic, OCTAVE-FLS denumit GNU Octave este o alternativă open-source la MATLAB, care include unelte dedicate dezvoltării de sisteme fuzzy. Octave nu are unelte incluse precum MATLAB, dar există pachete software dedicate pentru implementarea de sisteme fuzzy, care sunt susținute de Octave. Deoarece Octave este compatibil cu MATLAB, acesta susține sisteme fuzzy asemănătoare, adică Mamdani, Sugeno și sisteme hibride. Luând în considerare că Octave este open-source, utilizatorii pot să modifice și extindă sistemele fuzzy prin implementarea de caracteristici adiționale pentru a obține sisteme fuzzy de tipul 2.

Octave oferă următoarele avantaje utilizatorilor săi: gratuitate (open-source), compatibilitate cu MATLAB, personalizare, integrarea cu algoritmi de optimizare și alte tehnici IA și dezvoltarea de sisteme fuzzy bazate pe cod sursă.

La fel ca celelalte instrumente dedicate pentru dezvoltarea sistemelor fuzzy, și Octave are puncte slabe precum: lipsa unei interfețe grafice cu utilizatorii, nu are o unelată dedicată, ci trebuie instalate pachete suplimentare, folosirea limitată în industrie și dificultatea de integrare cu sisteme de control în timp real.

Limbajul de programare open-source Phyton nu are o unealtă inclusă predefinită precum MATLAB sau LabVIEW, dar suportă dezvoltarea și implementarea sistemelor fuzzy prin intermediul unor librării externe. Cea mai populară și utilizată librărie dedicată logicii fuzzy este scikit-fuzzy (skfuzzy), care oferă unelte pentru constuirea, simularea și analiza sistemelor fuzzy. Phyton prin intermediul librăriei scikit-fuzzy suportă următoarele sisteme fuzzy: Mamdani, sisteme fuzzy Sugeno (caracteristici limitate), sisteme fuzzy de tipul 2 și sisteme fuzzy hibride.

Avantajele utilizării Phyton și a librăriilor dedicate sunt: gratuitate, oferă personalizare ridicată, integrarea altor tehnici de IA și a algoritmilor de optimizare, flexibilitate în implementare și posibilitatea de a crea ușor prototipuri

și sisteme fuzzy pentru cercetare. Contrar acestor puncte forte, utilizarea Phyton pentru dezvoltarea de sisteme fuzzy aduce patru dezavantage:

- Lipsa unei interfețe grafice cu utilizatorul;
- Aplicații limitate în industrie;
- Necesitatea implementării sistemelor fuzzy sub formă de linii de comandă, ceea ce aduce nevoia unor cunoștințe de specialitate în programare în Phyton;
- Probleme de performanță – în aplicațiile în timp real, Phyton nu dă rezultate la fel de rapide precum alte limbaje de programare, precum C++.

Juzzy este o librărie open-source a limbajului de programare Java, care conține unelte dedicate pentru dezvoltarea sistemelor fuzzy. Funcțiile și uneltele Java din Juzzy sunt folosite intens în cercetarea academică, aplicațiile ale IA și aplicațiile de tip suport decizional. Acest lucru este încurajat și de faptul că Juzzy susține dezvoltarea de sisteme fuzzy de tipul 1, tipul 2, tipul 2 general și sisteme adaptive fuzzy și hibride.

Capabilitatea Juzzy de a susține un număr mare de tipuri de sisteme fuzzy este un mare avantaj, la care se adaugă alte puncte forte precum gratuitate, modularitate, interfață grafică cu utilizatorii și posibilitatea de a rula în Windows, macOS și Linux.

Dezavantajele Juzzy sunt:

- Utilizarea limitată în industrie;
- Dependență de Java;
- Comparativ cu FDL, Juzzy este mai puțin focalizat pe interfața cu utilizatorul;
- Nu suportă în mod implicit sistemele fuzzy funcționale în timp real.

Alte instrumente informatice care se pot folosi pentru dezvoltarea de sisteme fuzzy sunt limbajele de programare convenționale precum C și C++. Pentru o înțelegere mai clară, tabelul 4.4 ilustrează o comparație compactă a instrumentelor software prezentate pentru dezvoltarea de sisteme fuzzy. Caracteristicile care sunt luate în considerare în comparație sunt costul, ușurința în utilizare și sisteme fuzzy suportate.

**Tabelul 4.4.** Comparație între tehnicile de implementare a sistemelor fuzzy

| Atribut | MATLAB Fuzzy Logic Designer | LabVIEW Fuzzy Logic Toolkit | fuzzyTECH | Octave (Fuzzy Logic) | Python (scikit-fuzzy) | Juzzy (Java) |
|---|---|---|---|---|---|---|
| **Cost** | Preț ridicat | Preț moderat | Preț ridicat | Gratuit | Gratuit | Gratuit |
| **Ușurința în utilizare** | Mare | Mare | Mare | Medie | Medie | Medie |
| **Tipuri de sisteme fuzzy** | Tip-1, Sugeno, Mamdani, Hibrid | Tip-1, Mamdani, Sugeno | Tip-1, Mamdani, Sugeno | Tip-1, Mamdani, (Sugeno limitat) | Tip-1, Mamdani (Sugeno limitat) | Tip-1, tip-2, Mamdani, Sugeno |
| **Sisteme fuzzy tip 2** | Nu suportă inițial | Nu suportă în mod natural | Limitat | Nu suportă în mod natural | Parțial | Da |
| **Utilizarea în industrie** | Mare (Cercetare & industrie) | Mare (Industrie) | Mare (Automatizare industrială) | Mică (Cercetare/Educație) | Medie (Cercetare, IA, ML) | Mică (Cercetare) |
| **Interfața cu utilizatorul** | Da | Da | Da | Nu | Nu | Parțial |
| **Scriere de cod sursă** | Nu | Nu | Nu | Da | Da | Da |
| **IA & Alte tehnici** | Parțial | Nu | Nu | Nu | Da | Da |
| **Suportă sisteme integrate** | Da (cod sursă C) | Da (LabVIEW RT, FPGA, PLC) | Da (PLC, microcontrolere, cod sursă C) | Nu | Limitat | Nu |
| **Execuție în timp real** | Da (Simulink) | Da | Da | Nu | Nu | Nu |
| **Dedicat pentru** | Cercetare academică, inginerie, sisteme de control | Automatizare industrială, control în timp real | Control fuzzy industrial, sisteme integrate | Utilizare în educație, Open-Source Alternativă MATLAB | IA, ML, optimizare, cercetare | Sisteme de tip-2 avansate, cercetare |

4.4.2 Sisteme fuzzy fizice

Sisteme fuzzy hardware-based sunt implementate prin utilizarea unor dispozitive dedicate, cu scopul de a crește performanța și rapiditatea sistemelor fuzzy pentru aplicațiile în timp real. Cele mai populare implementări hardware sunt prin utilizarea microprocesoarelor (Arduino, RasberryPI, STM32), a FPGA (Field Programmable Gate Arrays) și a cipurilor fuzzy analog-digitale.

Această abordare oferă următoarele avantaje:

- Procesare rapidă în timp real;
- Eficient în controlul proceselor;
- Eficiență energetică;

- Dedicat unor aplicații specifice, dar mai puțin flexibil.

Aceste avantaje explică utilizarea lor intensă în industrie și aplicațiile comerciale, comparativ cu cele bazate pe software, care sunt în special dezvoltate de cercetători.

FPGA-urile oferă posibilitatea procesării paralele a regulilor fuzzy și mai mult, sistemul fuzzy implementat poate fi optimizat pentru creșterea vitezei de funcționare în aplicațiile în timp real. Sistemele fuzzy sunt implementate pe FPGA prin utilizarea VHDL sau Verilog.

Etapele în dezvoltarea sistemelor fuzzy prin implicarea FPGA-urilor sunt următoarele:

1. Definierea funcțiilor de apartenență și a regulilor fuzzy prin utilizarea MATLAB sau VHDL.
2. Conversia sistemelor fuzzy în logica digitală – întrucât FPGA-urile lucrează cu cu logica binară, toate datele fuzzy trebuie transformate în valori digitale:
    a. Se folosesc convertoare analog-digitale pentru a citi datele din sistemele reale;
    b. Implementarea funcțiilor de apartenență prin folosirea comparatorilor și a LUT (lookup table) în interiorul FPGA.
3. Implementarea inferențelor fuzzy în FPGA:
    a. Utilizarea regulilor de tipul IF..., THEN...., prin folosirea VHDL/Verilog;
    b. Implementarea metodei de rezolvarea a inferențelor (exemplu Max-Min);
    c. Implementarea unei metode de defuzzyficare.
4. Testare și optimizare.
    a. Realizarea unui model MATLAB, simularea și testarea lui;
    b. Implementarea în FPGA și testarea în timp real prin folosirea unor semnale electrice reale;
    c. Optimizarea pentru viteză.

Microprocesoarele sunt o variantă mai ieftină în realizarea sistemelor fuzzy fizice față de FPGA-urile, care sunt costisitoare. Dar, față de acestea din urmă, microprocesoarele oferă soluții mai lente, deci sunt mai puțin adecvate pentru controlul în timp real.

Implementarea sistemelor fuzzy prin microprocesoare presupune mai multe etape, dintre care primul se referă la alegerea tipului de microprocesoare ce urmează a fi folosite, deoarece acestea influențează modul în care sistemul fuzzy va fi implementat (diferite microprocesoare folosesc diferite limbaje de programare pentru introducerea codului de program). A doua etapă este scrierea componentelor sistemului fuzzy în limbajul acceptat de microprocesor, precum C sau C++. Astfel, se introduce partea de fuzzyficare, regulile, metoda de rezolvare a inferențelor și metoda de defuzzyficare. Următorul pas este realizarea sistemului fizic prin conectarea microprocesorului la senzori, traductoare, actuatorilor și alte elementele de execuție. Ultimul pas este testarea și optimizarea sistemului fuzzy dezvoltat. Testarea se face prin compararea cu varianta digitală a sistemului fuzzy.

Cipuri fuzzy analog-digitale sunt componente electronice specializate pentru realizarea directă de operații din logica fuzzy. Ele sunt compuse din circuite analoge (fuzzyficare și inferențe) și circuite digitale (evaluarea regulilor și defuzzyficare). Avantajele utilizării acestor dispozitive sunt viteza foarte mare de lucru, consum redus de putere și fiabilitate.

Un cip fuzzy în mod uzual conține următoarele componente: unitatea de fuzzyficare (convertește semnalele de la intrare în funcții de apartenență prin folosierea unor niveluri analoge ale tensiunii), motorul de inferență (implementează regulile fuzzy prin intermediul unui multiplicatoare analoge și a unor circuite sumatoare), unitatea de defuzzyficare (oferă la ieșire o valoare singulară analogă sau digitală), unitatea de memorie (stochează regulile predefinite în memoria ROM).

Dezvoltarea unui sistem fuzzy prin utilizarea cipurilor fuzzy presupune parcurgerea următorilor pași:

1. Alegerea unui cip fuzzy;
2. Realizarea interfeței cu dispozitivele care oferă datele de intrare;
3. Implementarea sistemului fuzzy pe cipuri;
4. Conectarea la actuatori;
5. Testarea și optimizarea.

O comparație între cele trei abordări în dezvoltarea sistemelor fuzzy fizice este prezentată în tabelul 4.5.

**Tabelul 4.5.** Comparație între FPGA, microprocesoare și cipuri fuzzy

| **Atribut** | **FPGA** | **Microprocessor /Microcontroller** | **Cip fuzzy** |
|---|---|---|---|
| **Viteza de procesare** | Foarte rapidă (procesare paralelă) | Moderată (Execuție secvențială) | Ultra-rapidă (Hardware-based) |
| **Performanța în timp real** | Excelentă (Răspuns la nivel de ns) | Bună (Răspuns la nievl de ms) | Cea mai bună (Răspuns la nivel de μs) |
| **Flexibilitatea în programare** | Mare (VHDL/Verilog) | Mare (C/C++) | Mic (Logică predefinită) |
| **Complexitatea implementării** | Mare (Necesită codare HDL) | Moderate (Necesită codare C) | Mic (Structură predefinită) |
| **Consum de putere** | Moderat - Mare | Mic - Moderat | Foarte mic |
| **Cost** | Mare | Mic spre moderat | Mic |
| **Scalabilitate** | Mare (suportă sisteme fuzzy mari) | Moderată (limitată de viteza CPU) | Mică (Număr de reguli fix) |
| **Adaptabilitate** | Reprogramabil (flexibil) | Programabil (flexibil) | Fix - neprogramabil |
| **Dedicate pentru** | **Sisteme fuzzy de dimensiuni mari și viteză mare** | **Sisteme integrate & control fuzzy adaptiv** | **Aplicații ultra-rapide, sisteme de decizie** |
| **Exemple în energetică** | **Detecția defectelor pe liniile de transport** | **Managementul rețelelor inteligente** | **Protecția la supracurenți prin relee** |

Mai multe informații se pot găsi în următoarele surse bibliografice [5] – [26].

## 4.5. Întrebări și exerciții

### Întrebări

1. Tipurile de sisteme fuzzy sunt
   Sisteme Mamdani / Sisteme Takagami / Sisteme fuzzy fizice

2. Controlul fuzzy dă rezultate mai slabe decât controlul PID
   Adevărat / Fals / Nu se aplică

3. Implementarea sistemelor fuzzy se poate face
   Analog / Digital / prin MATLAB

**Exerciții**

1. Fuzzyficarea unor mărimi fizice
   Se va considera o mărime electrică, care se va fuzzyfica prin folosirea a patru valori lingvistice și numere fuzzy corespunzătoare.

2. Luarea deciziei pentru reguli ca în exemplele date
   Se vor considera patru reguli și valoriile funcțiilor de apartenență corespunzătoare; apoi se va determina rezultatul folosind metoda Min-Max (Max-Min), Max-Prod și Sum Prod.

3. Scrierea regulilor pentru un sistem fuzzy de control
   Să se considere un sistem fuzzy de control din energetică pentru care se scriu regulile de inferență dintre mărimile de intrare și ieșire.

4. Defuzzyficarea unei suprafețe fuzzy oarecare.
   Se consideră o suprafață fuzzy oarecare (pentru ușurință se recomnadă să se folosească segmante de dreaptă – numere fuzzy triunghiulare, trapezoidale).

# 5

# Aplicații ale sistemelor fuzzy în managementul energiei

Acest capitol continuă tema sistemelor fuzzy, dar merge un pas înainte întrucât se focalizează pe prezentarea unor aplicații ale logicii fuzzy în managementul energiei propuse în literatura de specialitate.

## 5.1.Introducere

Logica fuzzy și sistemele fuzzy pot fi folosite cu succes în rezolvarea problemelor conexe proceselor energetice caracterizate de un anumit grad de incertitudine. Așa cum s-a prezentat în capitolul precedent, sistemele fuzzy sunt folosite cel mai frecvent în control, dar au fost utilizate cu succes și în predicție, luarea decizilor, recunoașterea modelelor și optimizare. Într-adevăr, sistemele fuzzy au fost implicate în managementul energiei în următoarele situații:

- Modelare - Calculul pierderilor de tensiune și de putere, determinarea parametrilor de rețea, estimarea sarcinilor electrice;
- Controlul proceselor industriale - Comanda comutatorului de ploturi sub sarcină la un transformator de putere, controlul unității cazan – turbină, controlul automat a debitului unui cazan cu aburi, reglarea temperaturii unor procese din energetică și industria materialelor de construcții, reglarea unor procese biologice;
- Controlul proceselor industriale - Reglarea proceselor de climatizare, reglarea unor procese legate de tracțiunea electrică, conducerea servo-sistemelor, controlul interacțiunii robot-mediu;
- Controlul proceselor neindustriale - Reglarea parametrilor legați de funcționarea mașinilor de spălat, aspiratoarelor, mașinilor de gătit, aparaturii electrocasnice, echipamentelor pentru încălzirea locuințelor, reglarea sistemelor de pază și alarmare;

- Controlul proceselor neindustriale - Controlul aparaturii electronice neindustriale, reglarea aparaturii optice, foto și video, controlul mijloacelor de transport auto și conexe.

Lista completă a tuturor tipurilor de aplicații din domeniul electroenergeticii a sistemelor fuzzy este:

1. Producerea de energie electrică
   - Distribuirea economică a consumului: logica fuzzy ajută la optimizarea producerii de energie, reducând în același timp costurile și luând în considerare incertitudinile legate de costurile și cererea de combustibil.
   - Unit Commitment: algoritmii bazați pe fuzzy decid ce generatoare ar trebui pornite/oprite, luând în considerare factori nesiguri, cum ar fi fluctuațiile cererii de consum.
   - Integrarea energiei regenerabile: controlerele fuzzy gestionează incertitudinea în sistemele de energie eoliană, solară și hibridă pentru a asigura o generare stabilă.
   - Controlul frecvenței: logica fuzzy ajustează ieșirea generatorului pentru a menține stabilitatea frecvenței în ciuda schimbărilor bruște ale cererii de la consumatori.
   - Control automat al producerii de energie electrică: Îmbunătățește răspunsul unităților de generare prin gestionarea incertitudinilor.
   - Controlul cazanelor în centrale termice: Sistemele fuzzy optimizează eficiența arderii prin gestionarea variațiilor raportului combustibil-aer.
   - Diagnosticarea defecțiunilor în centralele electrice: identifică și clasifică defecțiunile din centralele electrice pe baza recunoașterii modelelor fuzzy.
2. Transportul de energie electrică
   - Îmbunătățirea stabilității sistemului de alimentare: logica fuzzy ajută la stabilizarea tensiunii și frecvenței în timpul perturbărilor.
   - Controlul tensiunii și compensarea puterii reactive: controlerele bazate pe fuzzy reglează dispozitivele de compensare a puterii reactive (SVC, STATCOM) pentru a menține stabilitatea tensiunii.
   - Circulația de putere optimă: Optimizarea fuzzy este utilizată pentru a rezolva problemele OPF în condiții de încărcare incerte.
   - Controlul frecvenței: Controlerele fuzzy îmbunătățesc reglarea frecvenței în sistemele de transport buclate.

- Managementul congestionării transmisiei: metodele bazate pe fuzzy determină cea mai bună modalitate de a reduce congestia, menținând în același timp securitatea sistemului de alimentare.
- Protecția la distanță a liniilor de transmisie: releele pe bază de fuzzy îmbunătățesc detectarea și clasificarea defecțiunilor.
- Locația defecțiunilor în rețelele de transport: logica fuzzy estimează locația defecțiunilor folosind formele de undă de tensiune și curent.
- Evaluare dinamică a securității: evaluează securitatea sistemului în diferite condiții de operare folosind reguli fuzzy.

3. Distribuția energiei electrice
- Reconfigurarea rețelei de distribuție: logica fuzzy ajută la selectarea operațiunilor optime de comutare pentru a minimiza pierderile și pentru a îmbunătăți profilurile de tensiune.
- Detectarea, clasificarea și izolarea defecțiunilor: diagnosticarea defecțiunilor pe bază de fuzzy îmbunătățește fiabilitatea rețelelor de distribuție.
- Monitorizarea stabilității tensiunii: controlerele fuzzy gestionează generatoarele distribuite și bateriile de condensatoare pentru stabilitatea tensiunii.
- Gestionarea încărcării vehiculelor electrice: algoritmii fuzzy optimizează programele de încărcare a vehiculelor electrice pentru a reduce cererea de în perioadele de vârf de sarcină.
- Aplicații Smart Grid: logica fuzzy este utilizată pentru gestionarea cererii și luarea deciziilor în rețelele inteligente.
- Prognoza consumului: gestionează incertitudinea în predicția consumului pe termen scurt și pe termen lung.
- Restaurare serviciu: Ajută la restabilirea optimă a energiei după o întrerupere, luând în considerare constrângerile multiple.

4. Utilizarea energiei electrice
- Managementul energiei în clădiri: controlerele fuzzy optimizează sistemele HVAC, iluminatul și stocarea energiei.
- Controlul proceselor industriale: Controlerele pe bază de fuzzy reglează consumul de energie în mașinile industriale.
- Managementul răspunsului la cerere (demand response management): logica fuzzy ajută la gestionarea participării consumatorilor la programele de răspuns la cerere.

- Evaluarea calității energiei electrice: logica fuzzy este utilizată pentru a evalua golurile de tensiune, armonicile și alte probleme legate de calitatea energiei.
- Strategii de reducere a consumului: controlerele fuzzy decid cele mai bune strategii de reducere a consumului în condiții de urgență.
- Integrarea energiei regenerabile în microrețele: managementul energetic bazat pe fuzzy optimizează stocarea și generarea bateriilor de stocare în microrețele.
- Programarea aparatelor în case inteligente: sistemele fuzzy ajută la programarea utilizării aparatelor pentru a minimiza costurile cu energie.

În continuare, acest capitol prezintă aplicații ale logicii fuzzy din cele patru ramuri ale sistemelor electroenergetice, adică producere, transport, distribuție și utilizare, luând în considerare managementul energiei electrice, adică aplicațiile care abordă probleme de optimizare, eficiență energetică și control.

### 5.2. Aplicații în producerea energiei electrice

Sistemele fuzzy cu aplicații în partea de producere a energiei electrice pot fi utilizate în rezolvarea problemelor de integrare a energiei regenerabile. În această situație, logica fuzzy ajută la gestionarea incertitudinilor prin controlul sistemelor de stocare și ajustarea puterii sistemului energetic pentru a echilibra și contracara fluctuațiile caracteristice generării distribuite. În consecință, sistemele fuzzy ajută la menținerea stabilității sistemelor electroenergetice.

O aplicație a logicii fuzzy în managementul energiei, ramura de producere a energiei electrice este prezentată în articolul:

[27] Alves de Araujo Junior, C.A.; Mauricio Villanueva, J.M.; Almeida, R.J.S.d.; Azevedo de Medeiros, I.E. Digital Twins of the Water Cooling System in a Power Plant Based on Fuzzy Logic. *Sensors* 2021, *21*, 6737. https://doi.org/10.3390/s21206737.

Obiectivul studiului prezentat în articol a fost de a dezvolta gemeni digitale pentru un sistem de răcire cu apă cu echipamente auxiliare, în relație cu luarea deciziilor operatorului, pentru a determina numărul corect de ventilatoare. Acest model a fost dezvoltat pe baza algoritmului de extragere automată a regulilor fuzzy, derivat din sistemul SCADA unei centrale electrice situate în Paraíba, Brazilia. Gemenii digitali pot actualiza regulile sistemului fuzzy în cazul unor

evenimente noiprecum funcționarea în stare de echilibru, rampele de pornire și oprire și instabilitate.

Centrala electrică pentru care s-a realizat cercetarea, conține 40 de motoare cu ardere; iar fiecare motor este cuplat la un generator de 10 MVA, formând grupul motor-generator, care are o capacitate de livrare de 8,7 MW la ieșirea generatorului (valoare care derivă din puterea livrată la arborele motorului de 9 MW, din care se scad pierderile datorate cuplajului dintre motor și generator). Pentru a menține motorul în funcțiune pentru intervale lungi, este folosit un sistem de răcire cu apă care controlează temperatura aerului de alimentare (aerul de admisie după compresia în turbocompresor), temperatura uleiului de lubrifiere și temperatura capetelor. Pentru a crește eficiența unității de generare, toți parametrii de proces trebuie să fie controlați corespunzător, în special parametrii de temperatură, întrucât aceștia afectează direct funcționarea generatoarelor. Pentru a menține aceste niveluri de control, conform specificațiilor de funcționare, acest sistem de răcire consumă aproximativ 6 MW de energie pentru a-și menține puterea maximă.

Structura sistemului de răcire a apei este ilustrat în fig. 5.1. Centrala termoelectrică studiată în articol prezintă un sistem de răcire care folosește 55 de ventilatoare care sunt folosite pentru a controla temperatura apei de răcire a celor cinci motoare simultan la temperatura de referință de 62°C. Analiza sistemului de răcire a scos la iveală necesitatea dezvoltării unor gemeni digitali, cu ajutor cărora să se poată programa eficient operarea sistemulu. Sistemul digital are ca variabile de intrare următoarele: temperatura apei de intrare în radiatorul lateral de temperatură înaltă ($T_{in}$_HT), temperatura mediului ($T_{env}$), numărul de ventilatoare conectate, puterea generatorului Diesel 1F, puterea generatorului Genset Diesel 2, puterea Gensetului Diesel 3 (PowerDG3), puterea Gensetului Diesel 4 (PowerDG4) și puterea Gensetului Diesel 5 (PowerDG5), și variabilă de ieșire: temperatura apei de ieșire în radiatorul lateral de temperatură înaltă ($T_{out}$_HT). Acest set de variabile de intrare și ieșire a fost selectat pe baza cunoștințelor specialiștilor - profesioniști care au fost în contact zilnic cu operațiunile instalației și echipamentelor și bucla de control existentă la uzină [35].

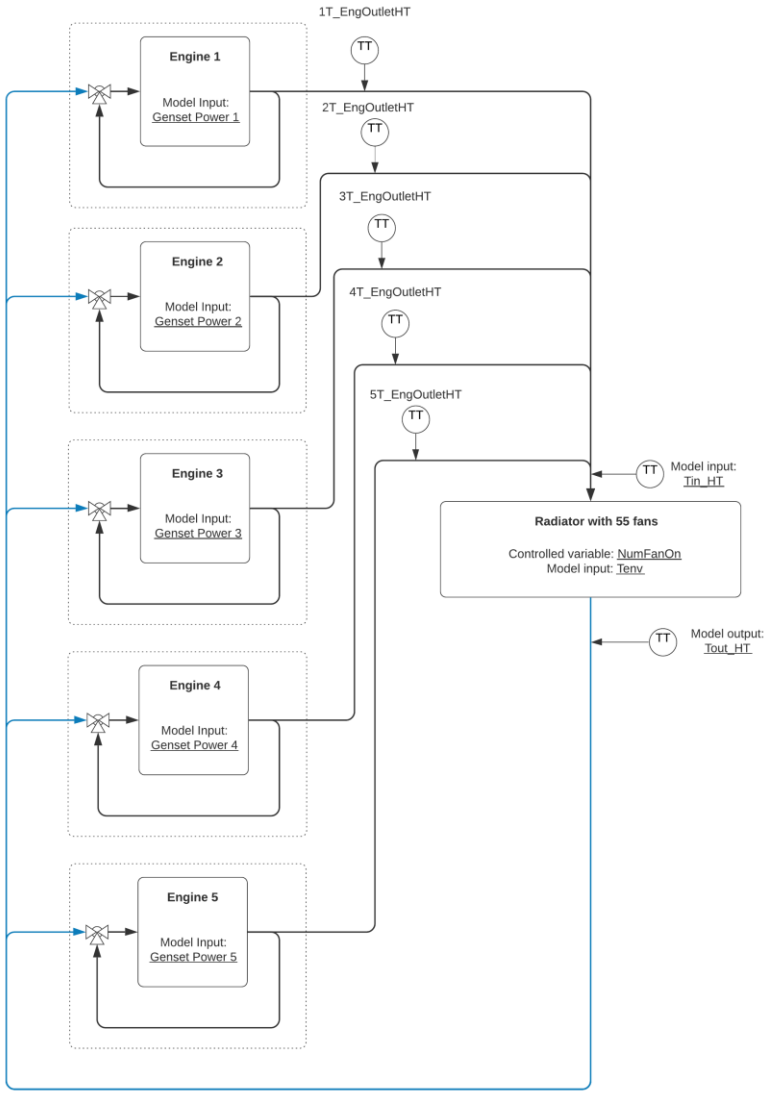



**Fig. 5.1.** Sistemul de răcire a apei [27]

Diagrama bloc a gemeni digital este prezentată în fig. 5.2. Sistemul de inferență fuzzy este un regulator de temperatură, al cărui punct de referință este temperatura definită în funcție de parametrii indicați de producător, fiind de 62°C. Pentru acest controler, intrările sunt eroarea și variația erorii, iar ieșirea este variația numărului de ventilatoare, care trebuie utilizată pentru a actualiza numărul de ventilatoare. Fig. 5.3 ilustrează fuzzyficarea variabilelor sistemului de inferență.

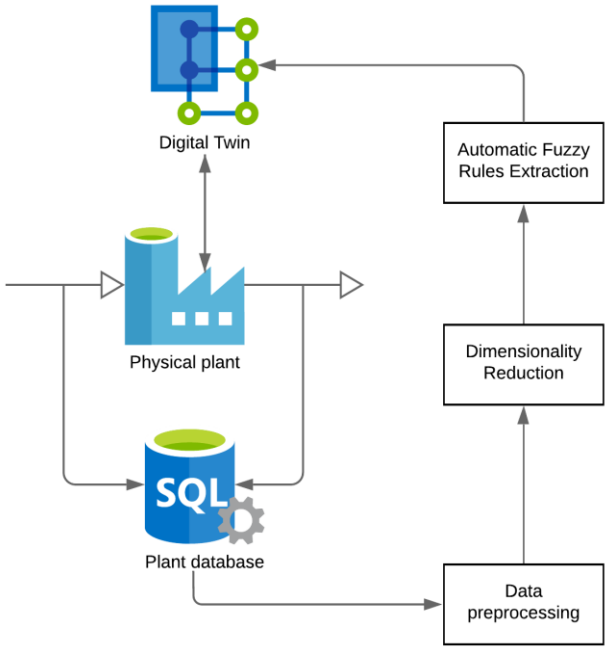



**Fig. 5.2.** Diagrama bloc a gemenilor digitali [27]

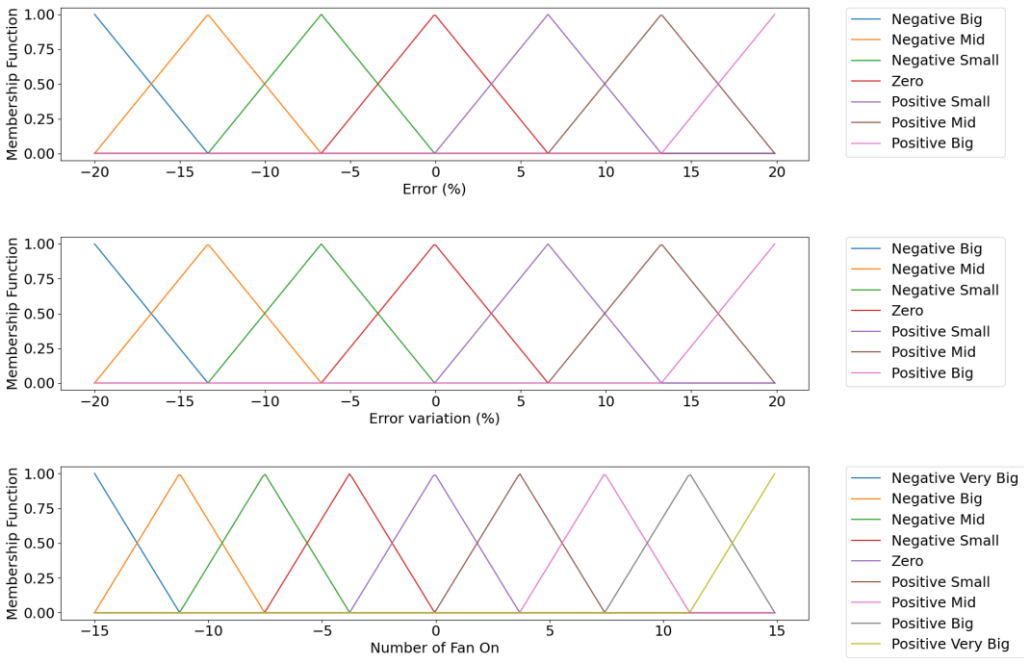



**Fig. 5.3.** Fuzzyficarea variabilelor sistemului fuzzy [27]

Sistemul fuzzy are 49 de reguli, dar agregarea regulilor și defuzzyficarea nu este clar cum s-a implementat. Testarea și validarea sistemului s-a realizat prin simulare și experimentare fizică.

O aplicație a logicii fuzzy în sistemele de producere a energiei electrice este prezentată în articolul:

[28] Magaji, N.; Mustafa, M.W.B.; Lawan, A.U.; Tukur, A.; Abdullahi, I.; Marwan, M. Application of Type 2 Fuzzy for Maximum Power Point Tracker for Photovoltaic System. *Processes* 2022, *10*, 1530. https://doi.org/10.3390/pr10081530.

În articol este prezentată o metodă bazată pe LF, de identificare a punctului maxim de producere a energiei electrice (MPPT – maximum-power point tracking) a unui sistem fotovoltaic. Mai mult, în studiu s-a comparat logica fuzzy de tip 2 (T2-FLC) cu metoda tradițională ”Perturb and Observe” (P și O) în trei scenarii diferite de iradiere, temperatură și factori de mediu, pentru a urmări punctul maxim de putere al fotovoltaicelor.

MPPT-ul bazat pe controler fuzzy, inspirat din conceptul de optimizare P și O, produce un răspuns de ieșire cu oscilație liberă. Fig. 5.4 și 5.5 oferă o reprezentare prin diagramă bloc a sistemului propus. Precum se observă, sistemul fuzzy dezvoltat a fost implementat folosind MatLab/Simulink. În fig. 5.5 este ilustrat controlul sistemului, care cuprinde controler fuzzy de tip 2, a căror date de intrare sunt prezentate în fig. 5.6.

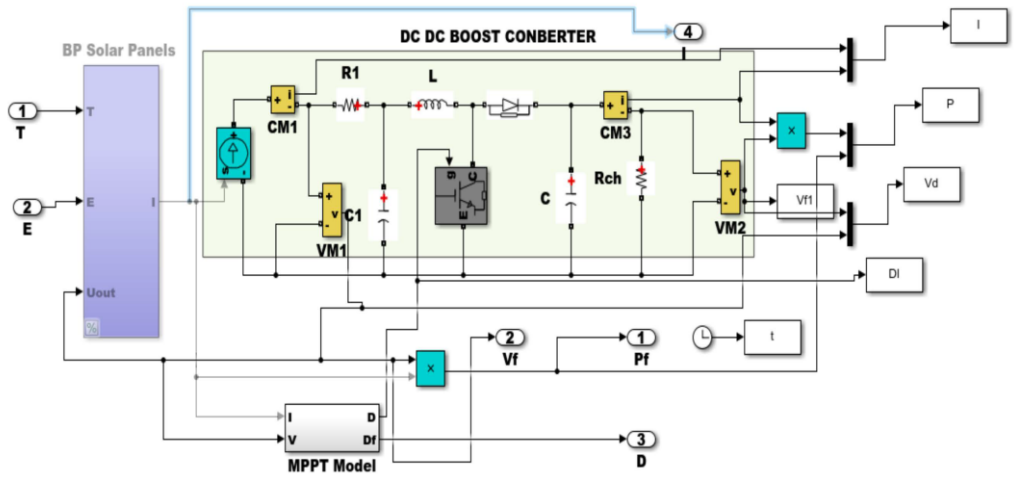



**Fig. 5.4.** Modelul Matlab a sistemului fotovoltaic cu MPPT [28]

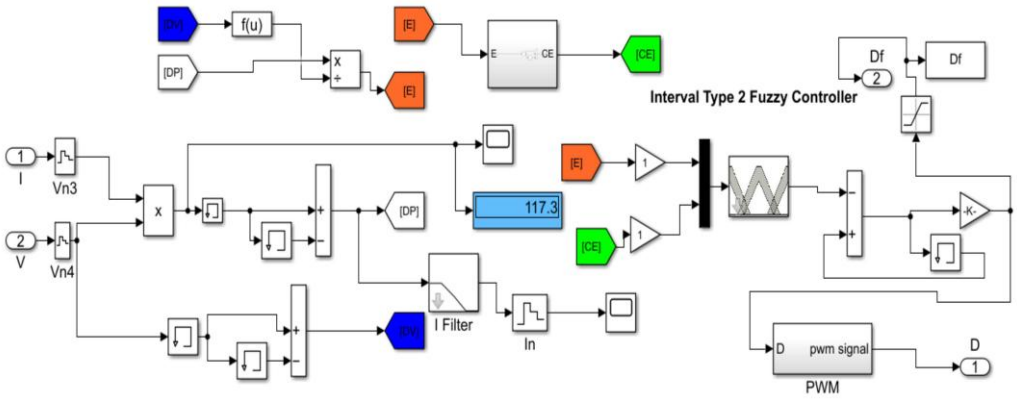



**Fig. 5.5.** Modelul Matlab a controlului MPPT [27]

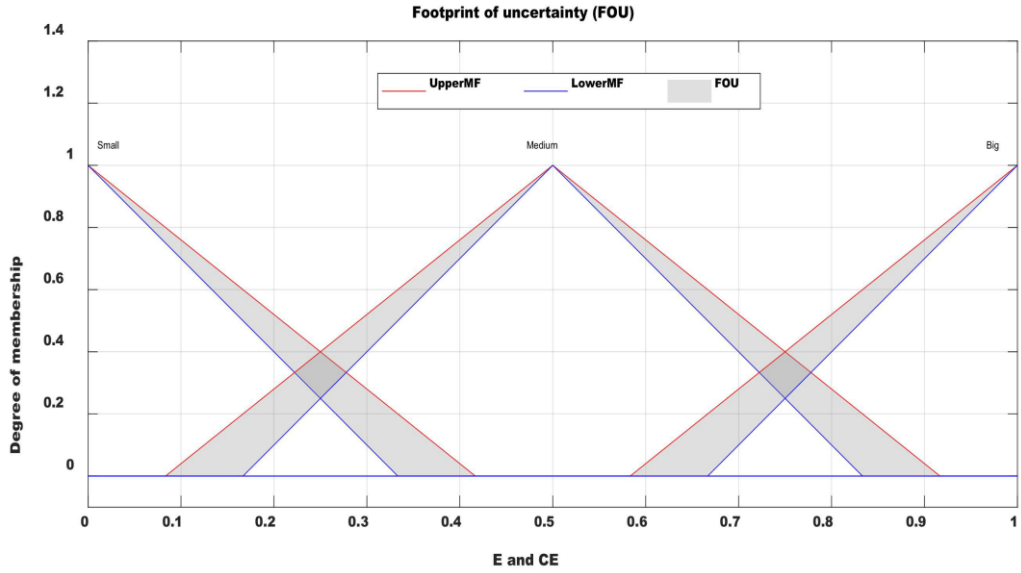



**Fig. 5.6.** Fuzzificarea datelor de intrare [28]

Datele de intrare au fiecare câte trei valori lingvistice, astfel sunt definite în articol nouă reguli. Metoda de defuzzyficare este centrul sumelor, methodă folosită după utilizarea modulului de reduce a numerelor fuzzy de tip 2 în numere fuzzy de tip 1.

Compararea rezultatelor obținute prin implicarea controlerului fuzzy de tip 2, cu rezultatele obținute prin metoda clasică fără controler, respectiv a unui

algoritm de optimizare, arată faptul că metoda propusă în articol este superioară celorlalte menționate.

### 5.3. Aplicații în transportul energiei electrice

În sistemele de transport al energiei electrice este necesară realizarea reglării tensiunii astfel încât aceasta să fie între limitele impuse prin standard. Sistemele fuzzy sunt folosite pentru controlul dispozitivelor precum SVC-uri și STATCOM-uri pentru a ajusta puterea reactivă și în consecință tensiunea în rețelele de transport. Astfel, prin controlul dinamic cu ajutorul controlerelor fuzzy, se pot combate fluctuațiile de tensiune și creșterea circulației de putere. Un articol care propune și prezintă o aplicație a logicii fuzzy în controlul tensiunii în rețelele de transport este:

[29] Rocha, P.H.A., Coelho, F. Fuzzy control for automatic operation of bank capacitors installed in a substation. *Electr Eng* 104, 4357–4366 (2022). https://doi.org/10.1007/s00202-022-01624-2.

Autorii propun o abordare bazată pe controlul fuzzy pentru operarea automată a bateriilor de condensatoare montate într-o stație de transformare, pe bara de 138 kV din Brazilia.

Sistemul de control prezentat în articol, așa cum se poate observa în fig. 5.7, este compus din 4 module.

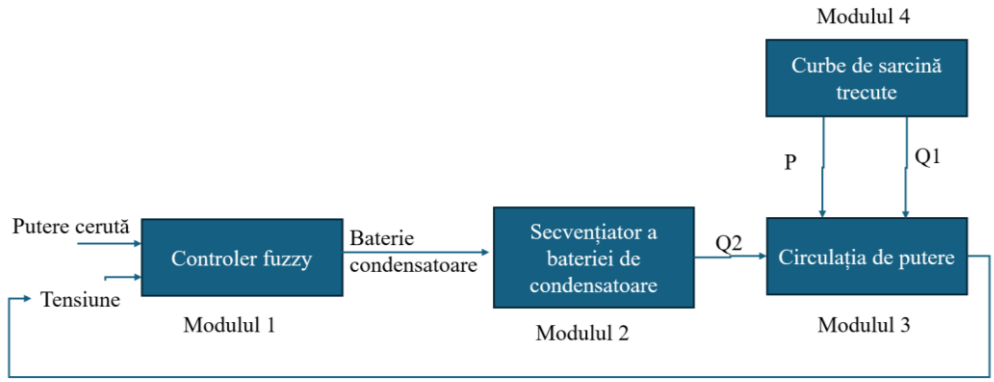



**Fig. 5.7.** Arhitectura sistemului de control [29]

În figură se poate observa că regulatorul fuzzy este responsabil pentru funcționarea bateriei de condensatoare, și primește drept intrare valoarea tensiunii de pe bara stației, și puterea necesată consumatorilor. În funcție de datele de la intrare, controlerul fuzzy oferă acțiunea care trebuie să die efectuată

asupra bateriei de condensatoare, care poate însemna: pornit, oprit sau nu acționa asupra bateriei de condensatoare. Această acțiune, notată BC, este trimisă secvențiatorului bateriei de condensatoare, care în funcție de logica de prioritate și acțiunea definită va determina o susceptanță. La final, blocul de circulație de putere, care simulează comportamentul substației primește date memorate cu privire la curba de sarcină a puterii active și reactive, precum și acțiunea bateriei de condensatoare de la modulul 2. Aceste date sunt folosite de modulul 4 pentru a calcula tensiunea pe bara stației.

Rolul controlerului fuzzy din modul 1 este de a menține valoarea tensiunii la valoarea maximă în timpul perioadelor de vârf de sarcină și de a reduce numărul de acțiuni realizate de bateria de condensatoare în perioadele de gol de sarcină.

Controlerul fuzzy este un sistem de tip Mamdani cu două date de intrare și una de ieșire. Fuzzyficarea datelor de intrare și ieșire s-a realizat în felul următor:

- Tensiunea are intervalul de variație 138-147 kV, care a fost împorțit în șapte subintervale cărora li se atribuie șase valori lingvistice, VL, L, ML, MH, H, VH. Valoarea lingvistică VL și VH sunt caracterizate de numere fuzzy treaptă, iar L, ML, MH și H au asociate numere fuzzy triunghiulare. Interesant este faptul că, valorile lingvistice de la cele două extremități, adică VL și VH sunt definite prin numere fuzzy care nu se intersectează cu celelalte din interiorul intervalului de confidență. Deci, intervale de incertitudine sunt doar între L, ML, MH și H.
- Variabila lingvistică Puterea cerută este definită pe intervalul 800-1100 MW. Pentru aceasta sunt definite trei valori lingvistice ”LIGHT”, ”MEDIUM” și ”HEAVY”. Numere fuzzy asociate valorilor lingvistice de pe marginea intervalului de confidență sunt din clasa numerelor treaptă-liniară, iar valoarea lingvistivă ”MEDIUM” are asociată un număr fuzzy triunghiular. Cele trei numere fuzzy care sunt folosite se suprapun, ceea ce este caracteristic logicii fuzzy, deci apar două intervale de incertitudine.
- Variabila lingvistică de ieșire, Capacitor Bank Operation – Funcționarea bateriei de condensatoare, este definită pe intervalul de confidență [0, 1]. Acestei variabile lingvistice îi sunt atribuite trei valori lingvistice, ”TURN OFF”, ”DO NOT OPERATE” și ”TURN ON”. Pentru cele trei valori lingvistice sunt folosite numere fuzzy trapezoidale, care se suprapun, deci sunt formate două intervale de icertitudine.

În legătură cu metoda de inferență și de defuzzyficare, nu se specifică nimic, în consecință, precum în cazul articolului precedent se presupune că a fost

folosită metoda Max-Min și centrul de greutate. Se cunoaște însă faptul că a fost folosit fuzzy logic toolbox de la MATLAB, iar suprafața de ieșire prezintă trepte în variația sa, care arată modul în care variabila de ieșire depinde de variabilele de intrare. Regulile dintre datele de intrare și ieșire sunt prezentate sub forma unei matrici de inferență.

Metoda propusă a fost implementată și testată într-o stație de transformare reală. Rezultatele testării a arătat că sistemul automat bazat pe logica fuzzy a oferit soluții mai bune decât operatorii umani, întrucât numărul de acțiuni ale bateriilor a scăzut, iar tensiunea s-a menținut în limitele impuse.

O altă aplicație este prezentată în articolul

[30] Khampariya, P.; Panda, S.; Alharbi, H.; Abdelaziz, A.Y.; Ghoneim, S.S.M. Coordinated Design of Type-2 Fuzzy Lead–Lag-Structured SSSCs and PSSs for Power System Stability Improvement. *Sustainability* 2022, *14*, 6656. https://doi.org/10.3390/su14116656.

Acest articol propune un sistem de reglare pentru controlere de amortizare bazate pe FACTS și stabilizatori ai sistemului de alimentare (PSS) pentru îmbunătățirea stabilității unui sistem electroenergetic. Mai exact, în articol se introduce o nouă structură de control, cunoscută sub numele de tip 2 fuzzy lead-lag (T2FLL), caracterizată prin controlerele bazate pe PSS și SSSC. În plus, se consideră că puterea cerută este o problemă de optimizare, iar parametrii controlerului sunt optimizați printr-o metodă hADE-PS propusă recent. Din punct de vedere tehnic, metoda hDE-PS este comparată cu metodele GA, PSO și DE, iar din punct de vedere al controlerului, T2FLL este comparată cu controlere de tip 1 fuzzy lead-lag (T1FLL) și controlere de lead-lag (LL) utilizate pe scară largă. Diverse scenarii de funcționare, respectiv locații de încărcare/defecțiuni sunt simulate atât pe magistrală principală (considerată cu sarcină infinită) pentru un singur generator (SMIB) cât și MMPS (mai multe generatoare). Structura unui SMIB este prezentată în fig. 5.8., unde transformatorul este T; tensiunile infinite ale magistralei și ale generatorului sunt *VS* și *VT*, iar sursa de tensiune continuă sunt reprezentate prin *Vcnv* și, respectiv, *VDC*; *V1* și *V2* sunt tensiunile pe bare, curentul de linie este *I*; puterile reale în liniile de transport și câte o linie sunt reprezentate fiecare de *PL* și, respectiv, *PL1*. Generatorul este prevăzut cu o turbină si regulator, un sistem de excitare, un PSS.

Regulatorul fuzzy propus are două intrări și o ieșire; el cuprinde blocul de fuzzyficare, regulile, rezolvarea inferenței, blocul de reducere și defuzzyficarea. Fuzzificarea s-a realizat prin folosirea unor numere fuzzy triunghiulare de tipul

2, câte cinci valori lingvistice pentru fiecare variabilă, 25 de reguli pentru inferență, rezolvarea inferenței nu este clară, dar pentru defuzzyficare s-au considerat trei metode: centru de greutate, centru sumelor și centrul suprafețelor. Structura celor două controlerelor considerate (în relație cu SSSC, respectiv în relație cu PPS) este ilustrată în fig. 5.9.

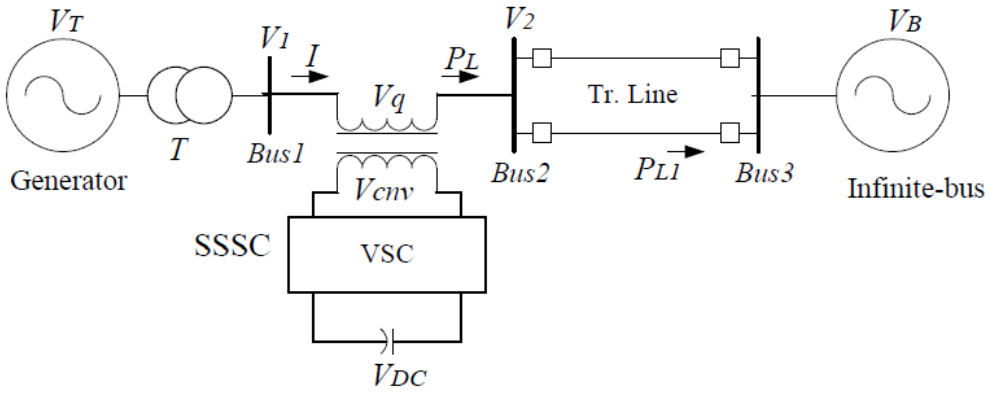



**Fig. 5.8.** Sistemul SMIB [30]

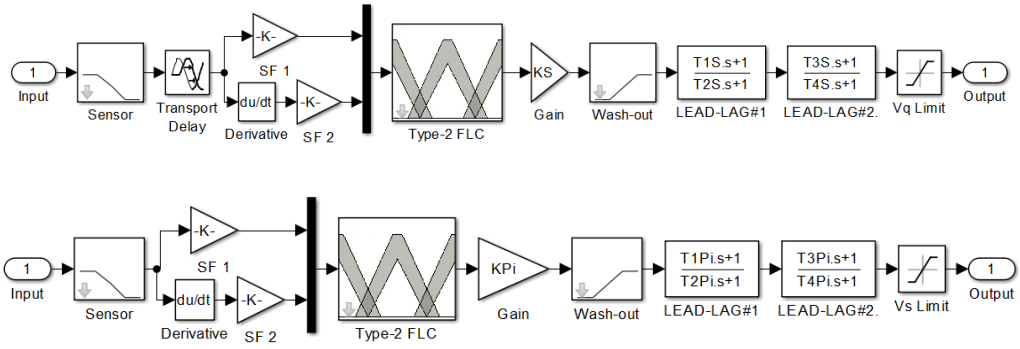



**Fig. 5.9.** Controler fuzzy bazat pe SSSC, respectiv pe PPS [30]

Implementarea metodologie propuse a fost realizată prin intermediul MatLab/Simulink, a cărui model este ilustrată în fig. 5.10. Acest model a fost testat, apoi sistemul de control a fost îmbunătățit printr-un algoritm de optimizare. Din nou, noile modele Simulink au fost implementate și testate. Acestea sunt descrise matematic și tehnic în articol.

Rezultatele obținute au arătat superioritatea metodei propuse față de metode similare din literatura de specialitate.

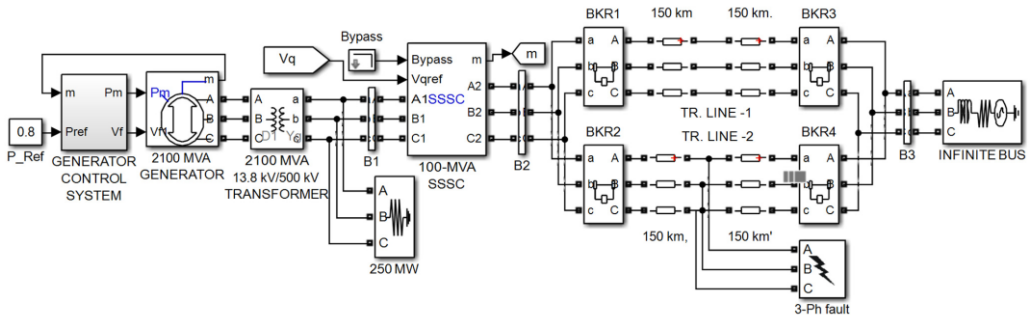



**Fig. 5.10.** Modelul Simulink a metodei propuse [30]

O aplicație care folosește controlul fuzzy adaptiv este prezentată în articolul:

[31] Rikos, E.; Caerts, C.; Cabiati, M.; Syed, M.; Burt, G. Adaptive Fuzzy Control for Power-Frequency Characteristic Regulation in High-RES Power Systems. *Energies* 2017, *10*, 982. https://doi.org/10.3390/en10070982.

În articol se specifică faptul că controlul sistemelor electroenergetice moderne va necesita folosirea pe scară largă a rezervelor de energie la nivel de distribuție a energiei electrice. Pentru a rezolva această problemă, studiu prezentat în articol propune o nouă strategie de ajustare dinamică a caracteristicilor frecvenței de putere bazată pe starea de dezechilibru. Controlul propus se bazează pe conceptul de „celule” care sunt sisteme generatoare cu capacități de operare și responsabilități similare zonelor de control, dar încurajând utilizarea resurselor la toate nivelurile de tensiune, în special rețelele de distribuție. În studiu, caracteristica frecvență-putere a unei celule este ajustată în timp real prin intermediul unui controler fuzzy. Această abordare duce la reducerea unei părți din rezerve, determinând o activare mai localizată și deci scăderea pierderile, congestiile și folosirea eficientă a rezervelor. Fig. 5.11 ilustrează conceptul de celulă în contextul unui sistem electroenergetic real. Precum se observă, sistemul după împărțire cuprinde cinci celule în funcție de nivelul tensiunii (un exemplu este celula de înaltă tensiune (HV cell)).

Strategia de control a metodei propuse este prezentată în fig. 5.12. În această diagramă răspunsul în frecvență al celulei *i* este reprezentat de funcția de transfer $G_{pi}(s)$ și implică parametrii constantei de inerție $H_i$, precum și autoreglarea sarcinii $D_i$.

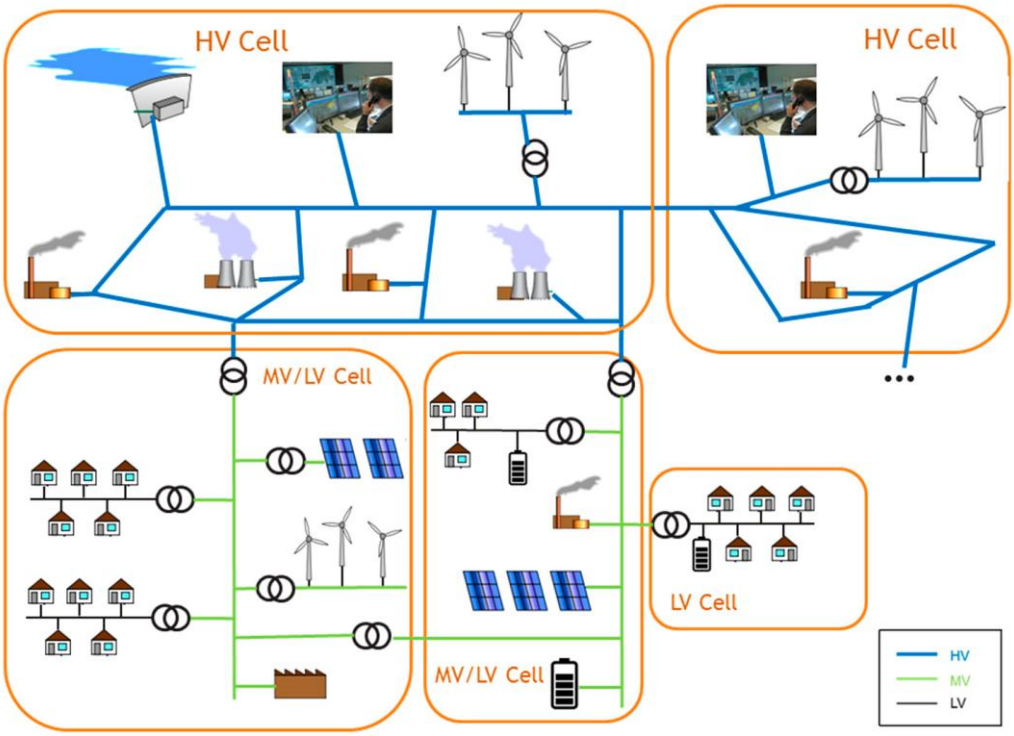



**Fig. 5.11.** Împărțirea sistemului electroenergetic în celule [31]

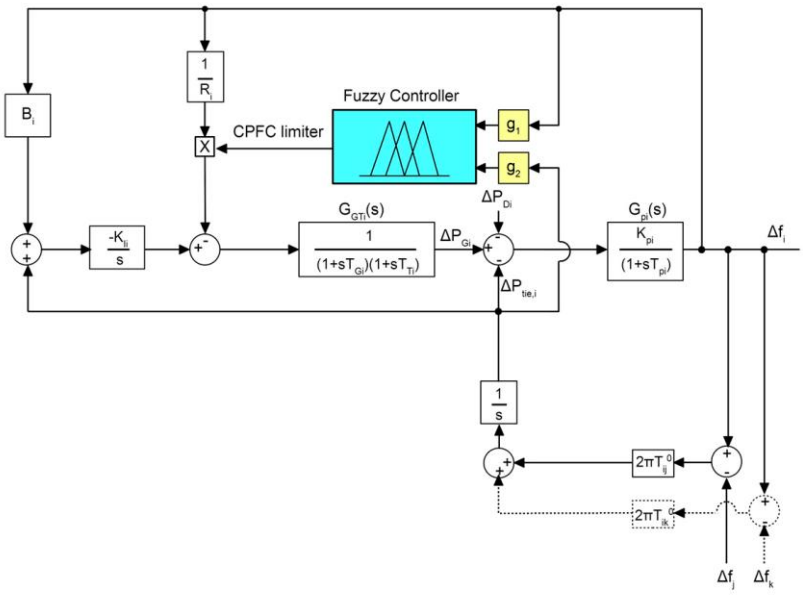



**Fig. 5.12.** Diagrama controlului propus implementat pentru o celulă *i* [31]

Sistemul fuzzy folosește câte cinci numere fuzzy triunghiulare pentru cele două date de intrare și variabila de ieșire. Mai mult, nu se știe tipul de sistem fuzzy folosit și metoda de rezolvare a inferențelor, dar metoda de defuzzyficare implementată este metoda centrului de greutate.

La fel ca în cazul articolelor precedente, și în această lucrare se demonstrează superioritatea metodei propuse comparativ cu alte metode.

### 5.4. Aplicații în distribuția energiei electrice

Logica fuzzy este folosită și în sistemele de distribuție a energiei electrice. O problemă des întâlnită în distribuția de energie electrică este oprirea alimentării cu electricitate în urma unui defect. În acest context, sistemele fuzzy ajută în identificarea secvenței optime de punere în funcțiune a aparatelor dedicate pentru a restaura alimentarea cu energie electrică în cel mai eficient mod, pentru a minimiza pierderile și costurile. Un articol științific care propune o metodă pentru analiza eșecului operatorilor din sistemele de distribuție din Ecuador este

[32] Dayana Andrade-Benavides, Diego Vallejo-Huanga, Paulina Morillo, Fuzzy Logic Model for Failure Analysis in Electric Power Distribution Systems, Procedia Computer Science, Volume 204, 2022, Pages 497-504, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2022.08.061.

Scopul studiului prezentat în articolul menționat este de a aplica modelele fuzzy și variabilele legate de timpii de deconectare în prezența diferiților agenți, și a determina gradul de severitate a defecțiunilor în continuitatea serviciului de alimentare cu energie electrică pentru o companie de distribuție.

Diagrama bloc a metodei propuse este ilustrată în fig. 5.13. Precum se observă din figură, patru factori (agenți) au fost luați în considerare ca posibile cauze a unei defecțiuni care să cauzeze oprirea alimentării cu energie electrică (eșecul), și anume:

- condiții climatice (vânt, ploaie, ninsoare, ceață, descărcări atmosferice și căldura solară);
- animale (păsări și rozătoare);
- accidente cauzate de terțe părți (accidente rutiere și alte accidente);
- rețeaua de distribuție (materiale și construcție de proastă calitate).

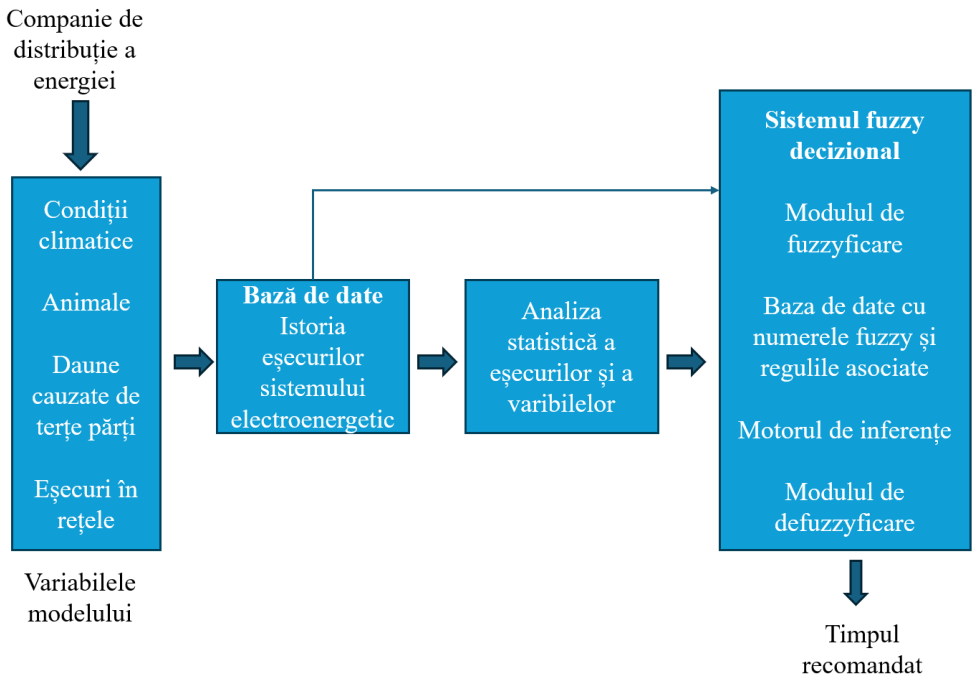



**Fig. 5.13.** Diagrama bloc a metodei propuse [32]

Datele de intrare ale sistemului fuzzy de evaluare a performanței companiei provin de la o bază de date care conține informații legate de istoria defecțiunilor apărute din cauza defectelor în sistemul electric, și de la o altă bază de date care cuprinde date statistice cu privire la variabilele și defecțiunile apărute. Se observă din diagramă, faptul că sistemul fuzzy conține un bloc de fuzzyficare, o bază de cunoștințe cu numerele fuzzy asociate datelor de intrare și ieșire și setul de reguli, un motor de inferențe și un modul de defuzzyficare. Modul de prezentare a sistemului fuzzy este diferit, dar în esență, el conține același tip de informații și funcționare. Sistemul fuzzy va oferi la ieșire o variabilă care indică eficiența în ceea ce privește timpul de reconectare a serviciului, de către compania de distribuție în caz de defecțiune.

Sistemul fuzzy are cinci variabile lingvistice cu care lucrează. Intrările sunt reprezentate de următoarele mărimi: condiții climatice, animale, defecțiuni cauzate de terțe părți și defecțiuni ale rețelei, iar ieșirea este timpul de reconectare (care cuantifică eficiența companiei de distribuție în rezolvarea defecțiunii și restabilirea alimentării cu energiei electrică). Datele de intrare au fost fuzzyficate prin folosirea a trei valori lingvistice pentru fiecare și a numerelor fuzzy trapezoidale pentru funcțiile de apartenență, respectiv a patru valori lingvistice

pentru ieșire. Aceste valori lingvistice au valorile: foarte eficient, eficient, slab, foarte slab.

Autorii specifică despre sistemul fuzzy că este de tip Mamdani, metoda de inferență folosită este Min-Max, iar defuzzyficarea prin aplicarea metodei centrului de greutate.

Sistemul a fost implementat, simulat și testat prin folosirea Phyton, mai exact a librăriei *scikit-fuzzy*. Pentru testare au fost simulate 16 scenarii, care au demonstrat eficacitatea sistemului propus și posibilitatea utilizării lui în situații reale. Rezultatul rezolvării inferențelor în situația în care variabila de ieșire TR este slab.

### 5.5. Aplicații în utilizarea energiei electrice

Sistemele fuzzy sunt folosite în casele inteligente pentru a programa funcționarea electrocasnicelor, cu scopul de a realiza economii de preț și consum de energie electrică. O aplicație propusă în literatura de specialitate este descrisă în articolul

[33] Q. -U. Ain, S. Iqbal and H. Mukhtar, Improving Quality of Experience Using Fuzzy Controller for Smart Homes, *IEEE Access*, vol. 10, pp. 11892-11908, 2022, doi: 10.1109/ACCESS.2021.3096208. [https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9481080].

În articol, autorii propun o abordare care presupune utilizarea într-o casă inteligentă (fig. 5.14.) a unui controler fuzzy, $FLC_{h1}$, care să optimizeze funcționarea unui dispozitiv HVAC, respectiv a unui al doilea controler fuzzy, $FLC_{l1}$ pentru a controla nivelul de iluminare; acestea cu scopul de a crește confortul și a obține eficiență energetică.

Controlerul fuzzy pentru reglarea nivelului de iluminare are structura descrisă în fig. 5.15. Se observă că, regulatorul are cinci date de intrare: iluminarea exterioară, iluminarea interioară, gradul de ocupare, prețul energiei electrice și prioritatea, iar ieșirea este nivelul de iluminare. Valoriile lingvistice și numere fuzzy asociate lor, cu funcțiile de apartenență corespunzătoare sunt descrise în fig. 5.16. Reiese din figură că, variabila lingvistică gradul de ocupare are doar două valori lingvistice, Absent și Present, cu numere fuzzy treaptă. De asemenea, numerele fuzzy folosite sunt din clasele numerelor fuzzy triunghiulare și trapezoidale.

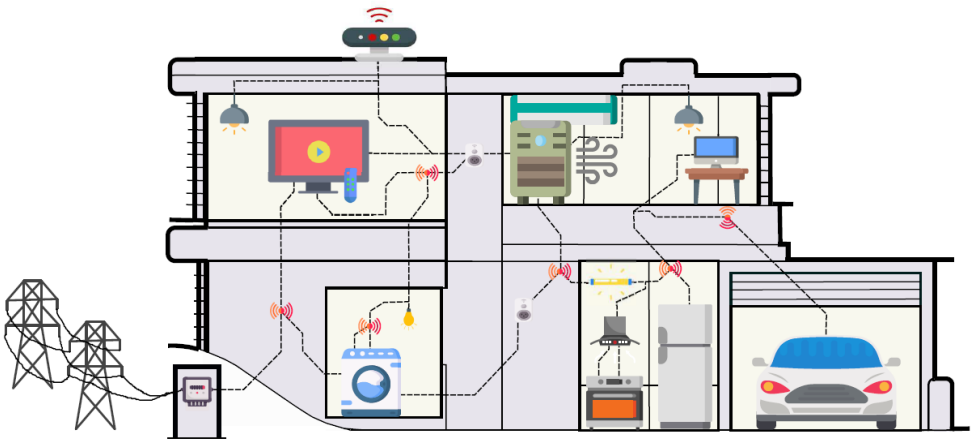


**Fig. 5.14.** Instalație electrică – casă inteligentă [33]

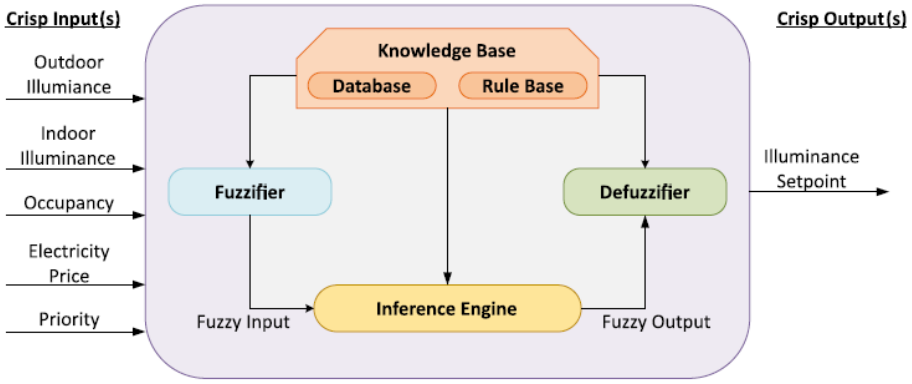



**Fig. 5.15.** Controlerul fuzzy pentru reglarea nivelului de iluminare [33]

Metoda de rezolvare a inferențelor este Max-Min, în cadrul unui sistem de tip Mamdani, iar determinarea valorii singulare de ieșire se determină folosind metoda centrului de greutate.

Controlerul fuzzy pentru reglarea aparatului HVAC este de tip Mamdani, cu 6 variabile de intrare și o ieșire, care indică consumul de energie a aparatului. Variabilele lingvistice de intrare sunt temperatura exterioară, temperatura interioară, prețul la momentul folosirii, ocuparea spațiului, umiditatea relativă și valoarea de referință a termostatului. În fuzzyficarea acestor variabile au fost folosite câte trei valori lingvistice și numere fuzzy triunghiulare și trapezoidale. Luând în considerare numărul de variabile și valori lingvistice, au fost definite 486 de reguli fuzzy. La fel ca în cazul controlerului precedent, rezolvarea

inferențelor a fost realizată prin metoda Max-Min, iar defuzzyficarea prin metoda centrului de greutate.

Implementarea și testarea sistemului fuzzy pentru casa inteligentă au fost realizate folosind MATLAB, iar rezultatele prezentate de autori au indicat eficiența sistemului.

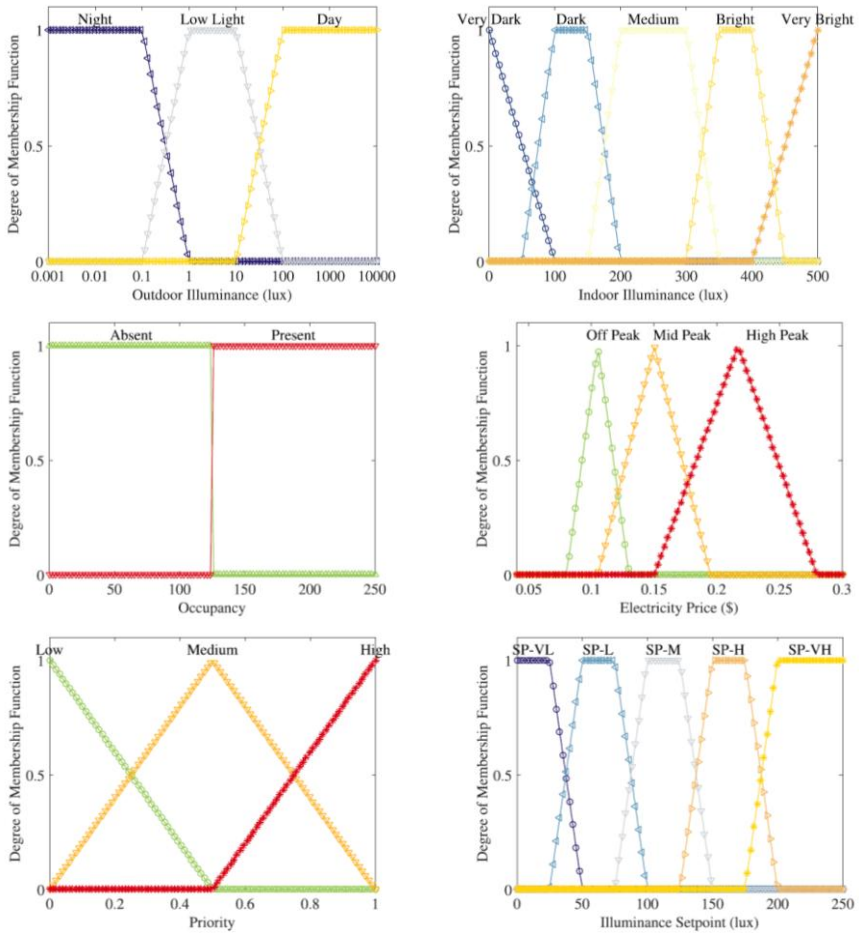



**Fig. 5.16.** Fuzzyficarea datelor de intrare și ieșire a controlerului [33]

O altă aplicație a logicii fuzzy este prezentată în articolul

[34] Tavarov, S.S.; Matrenin, P.; Safaraliev, M.; Senyuk, M.; Beryozkina, S.; Zicmane, I. Forecasting of Electricity Consumption by Household Consumers Using Fuzzy Logic Based on the Development Plan of the Power System of the Republic of Tajikistan. *Sustainability* 2023, *15*, 3725. https://doi.org/10.3390/su15043725.

Sistemul fuzzy propus în articol este un sistem de prognoză a consumului de energie electrică. Prognoza se bazează pe date statistice privind consumul de energie din perioada anterioară. Abordarea folosită în studiul prezentat în articol este o abordare probabilistică care permite prelucrarea evaluărilor experților oferite sub formă de declarații naturale folosind criterii calitative, formalizându-le și dând criteriilor o formă flexibilă, dar cantitativă.

Folosind datele consumului de energie se realizează fuzzyficarea acestuia, rezultând valorile lingvistice și numerele fuzzy din fig. 5.17, pentru anotimpul iarna.

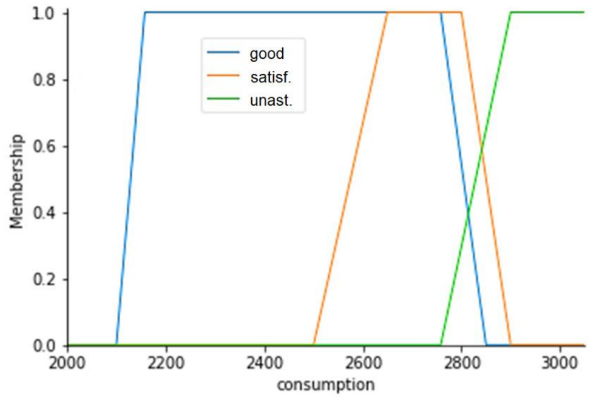



**Fig. 5.17.** Fuzzyficarea consumului de energie [34]

Variabilele lingvistice de intrare sunt indicatorii de intrare ai corespondenței schimbărilor lingvistice „Satisfacția consumatorului", „Eficiența modului de consum" și „Gradul de conformitate cu conceptul companiei". Fiecare dintre aceste variabile lingvistice au câte trei valori lingvistice, cărora le sunt asociate numere fuzzy gaussiene (fig. 5.18).

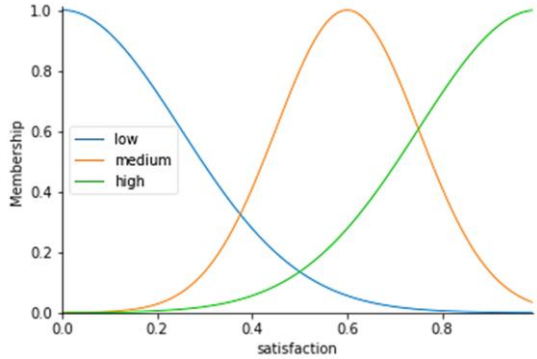



**Fig. 5.18.** Fuzzyficarea Satisfacției consumatorului [34]

Implementarea modelului fuzzy, inclusiv crearea de funcții de apartenență, a regulilor fuzzy, a algoritmului de inferență fuzzy al aplicației Mamdani și vizualizare au fost realizate folosind biblioteca open-source Python SciKit-Fuzzy (github.com/ scikit-fuzzy/scikit-fuzzy, accesat la 5 noiembrie 2022).

Rezultatele obținute prin rularea aplicației și explicarea acestora sunt foarte clar prezentate în fig. 5.19, pentru datele de intrare - consum 2200 kWh, satisfacție 0,5, randament plan 5,0, și în fig. 5.23 cu datele de intrare consum 2400 kWh, satisfacție 0,9, eficiență plan 7,1.

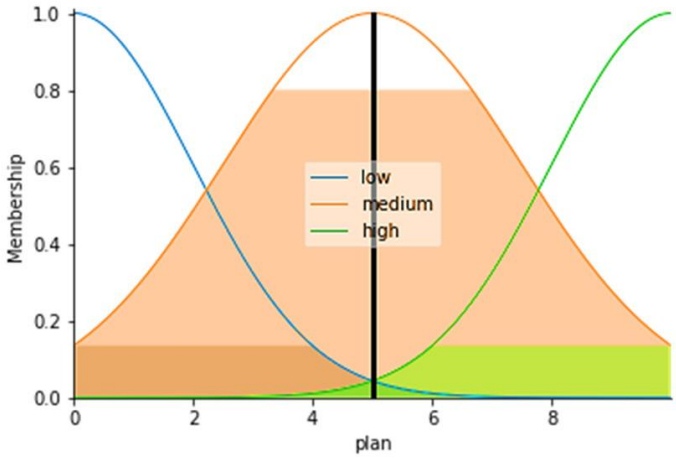



**Fig. 5.19.** Rezultatul obținut pentru date de intrare specifice [34]

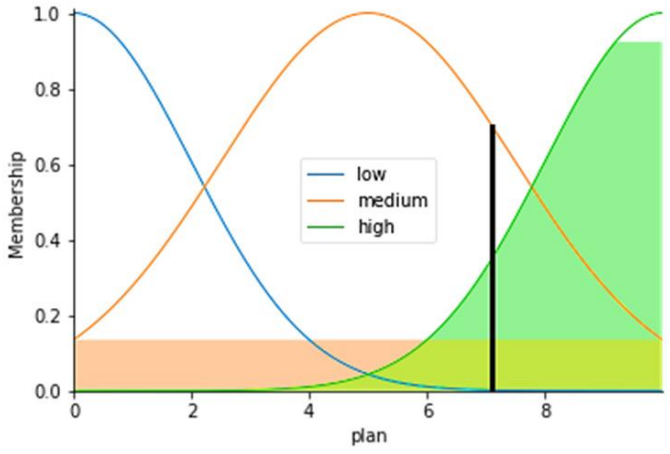



**Fig. 5.20.** Rezultatul obținut pentru date de intrare specifice [34]

Rezultatele prezentate în articol și pentru alte date de intrare, precum și compararea lor cu rezultatele obținute cu alte metode, subliniază eficiența metodei propuse.

O aplicație fuzzy în integrarea energiei regenerabile este prezentată în articolul:

[35] Lanjing Xu, Zhiwei Wang, Yanfeng Liu, Lin Xing, Energy allocation strategy based on fuzzy control considering optimal decision boundaries of standalone hybrid energy systems, Journal of Cleaner Production, Volume 279, 2021, 123810, ISSN 0959-6526, https://doi.org/10.1016/j.jclepro.2020.123810.

Articolul prezintă o strategie de alocare flexibilă a surselor de energie regenerabilă bazată pe un controler fuzzy în două trepte, cu scopul de a rezolva problemele de management al energiei și a controlului costurilor în sisteme electroenergetice autonome (fig. 5.21.). O atenție deosebită a fost acordată gestionării sistemelor de stocate și a resurselor de urgență (rezervă). Din figură reiese că sistemul energetic autonom conține trei unități generatoare (o turbină eoliană, un generator PV și un generator Diesel), un sistem de stocare (baterie), un chiller, controlerul (sistem de management a energiei), covertoare (AC/DC, DC/DC și DC/AC), o bară DC și conductoarele electrice necesare. Se poate observa că sistemul energetic conține: un singur receptor, chillerul, plus două generatoare cu producție variabilă, dependentă de factorii de mediu, un generator care se poate folosi la nevoie, un sistem de stocare și desigur unitatea centrală de control care reglează alimentarea receptorului.

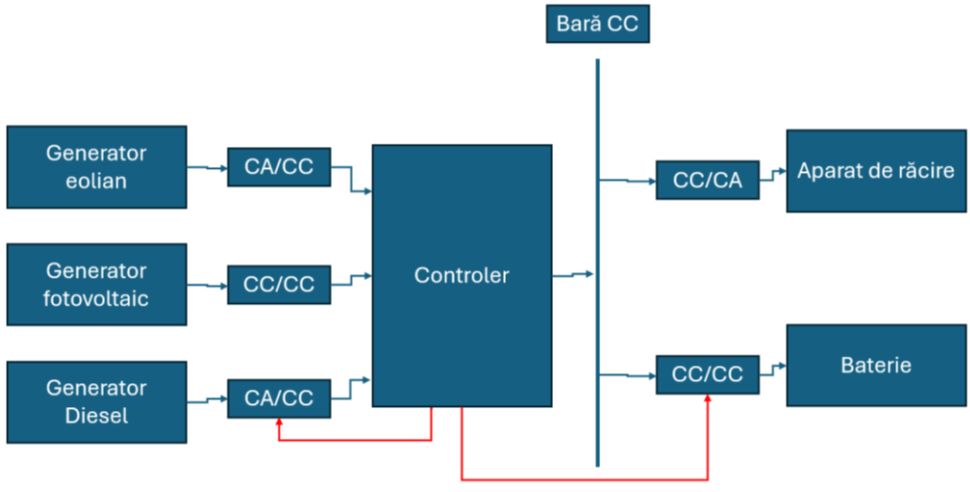



**Fig. 5.21.** Structura sistemului energetic autonom [34]

Comparativ cu un controler fuzzy convențional, în articol sunt introduse limite de decizie, care sunt optimizate pe baza caracteristicilor reale de funcționare ale unităților de generare. Se specifică în articol faptul că, controlerul fuzzy care ia în considerare limitele de decizie este mai obiectiv și realizează un management mai precis al energiei în timp real, cu costuri mai mici. Acest nou controler are performanțe mai bune în ceea ce privește fiabilitatea, economia și viteza de răspuns pentru un sistem în timp real. Strategia propusă are un efect pozitiv asupra aplicării practice a sistemelor hibride de energie regenerabilă pentru clădirile din zone izolate.

Modul de funcționare a sistemului fuzzy propus este ilustrat în fig. 5.22. Sistemul de management al energiei cuprinde două controlere fuzzy în cascadă. Primul controler FLC1 primește ca date de intrare puterea $\Delta P$ la pasul $t$, care reprezintă diferența dintre puterea produsă de generarea distribuită (generatorul fotovoltaic și generatorul eolian) și puterea receptorului, și starea bateriei la pasul $t$-1, iar ca ieșire este factorul de corecție $K_{bat}$ al comenzii originale de încărcare/descărcare a bateriei. Deci, rolul lui FLC1 este de a controla bateria, mai exact procesele de încărcare și descărcare ale acesteia. Al doilea controler fuzzy, FLC2 are rolul de a controla generatorul Diesel (notat în imagine DG) astfel încât să se asigure necesarul de putere pentru funcționarea receporului.

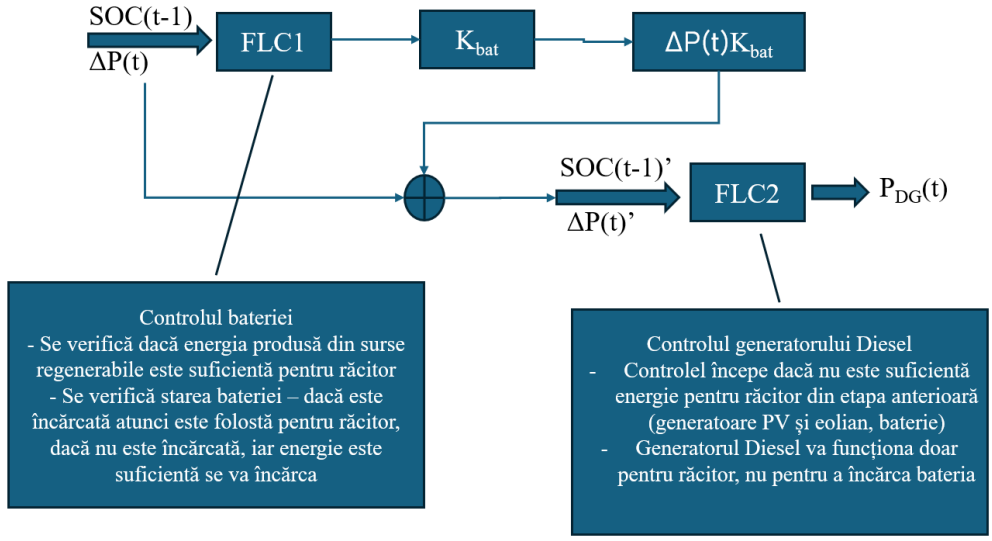



**Fig. 5.22.** Schema de funcționarea sistemului propus [34]

Datele de intrare ale lui FLC2 sunt starea bateriei după etapa 1 (funcționarea lui FLC1) și diferența de putere secundară (care ia în considerare și energia care

o poate asigura bateria). Ieșirea lui FLC2 este puterea care trebuie să o producă generatorul Diesel.

În concluzie, ideea principală a sistemului fuzzy de control a puterii este de a asigura alimentarea receptorului din energie regenerabilă. Dacă nu este suficientă energie produsă din surse regenerabile, atunci se va folosi energia înmagazinată în baterie în timpul orelor de vârf, când s-a produs energie electrică din energie regenerabilă în exces. Dacă nu este destulă energie înmagazinată, atunci se va folosi energia produsă de generatorul Diesel.

Sistemul de control a fost implementat, simulat și testat prin MATLAB/Simulink, diagrama funcțională din MATLAB este realizată pe baza schemei de funcționare din fig. 5.22. Pentru implementarea controlelor fuzzy a fost folosit Fuzzy Logic Toolbox. Caracteristicile controlerelor fuzzy sunt descrise în paragrafele următoare.

Controlerele FLC1 și FLC2 sunt sisteme fuzzy de tip Mamdani, cu două date de intrare și una de ieșire.

Regulatorul FLC1 folosește numere fuzzy triunghiulare și trapezoidale pentru a reprezenta atât datele de intrare cât și cea de ieșire. Pentru toate mărimile cu care lucrează controlerul sunt definite câte șapte valori lingvistice pentru fiecare variabilă lingvistică. Variabila de intrare ΔP are șapte numere fuzzy triunghiulare. Variabila de intrare SOC folosește șase numere fuzzy triunghiulare și un număr fuzzy trapezoidal central. Aceași reprezentare, prin implicarea a șase numere fuzzy triunghiulare și trapezoidale este folosită și pentru mărimea de ieșire $K_{bat}$.

Regulatorul FLC2 folosește numere fuzzy Gauss pentru a reprezenta atât datele de intrare cât și cea de ieșire. Pentru toate mărimile cu care lucrează controlerul sunt definite câte șapte valori lingvistice pentru fiecare variabilă lingvistică. Variabila de intrare SOC' are șapte numere fuzzy gaussiene egale. Variabila de intrare ΔP' folosește șapte numere fuzzy gaussiene de dimensiune variabilă. Aceași reprezentare prin implicarea a șapte numere fuzzy gaussiene egale este folosită pentru mărimea de ieșire $P_{DG}$.

Regulile de inferență a celor două regulatoare sunt descrise prin intermediul a două matrici de inferență.

Suprafețele fuzzy de ieșire pentru cele două controlere sunt prezentate în articol. Dar, nu se specifică metoda de rezolvare a inferențelor și cea de defuzzyficare. Însă, luând în considerare că a fost folosit fuzzy logic toolbox de la MATLAB, care folosește implicit metoda Max-Min pentru rezolvarea

inferențelor și metoda Centroid pentru defuzzyficare, atunci cel mai probabil că acestea au fost folosite.

Observație

Acest aspect, și anume lipsa mențiunii metodelor de defuzzyficare și de rezolvare a inferențelor, l-am întâlnit la aproximativ 40% din aplicațiile propuse în literatura de specialitate.

În literatura de specialitate este necesar ca atunci când se propune o nouă metodă, ca aceasta să fie comparată cu altă metodă recunoscută, pentru a justifica eficiența sistemului propus. Astfel, în acest articol, sistemul fuzzy propus a fost comparat (prin intermediul rezultatelor lui) cu un sistem fuzzy simplu (fără limite de decizie) și cu metoda convențională acceptată în literatură prin care se impune o prioritate fixă în modul în care este folosită energia produsă din diferite surse de energie.

## 5.6. Întrebări și exerciții

Întrebări

1. Sistemul fuzzy folosit în sistemele de producere a energiei electrice este un sistem Sugeno.
   Adevărat / Fals

2. Sistemul fuzzy folosit în sistemele de transport a energiei electrice are drept rol controlul tensiunii.
   Adevărat / Fals

3. Sistemul fuzzy folosit în sistemele de distribuție a energiei electrice, este un sistem fuzzy de control.
   Adevărat / Fals

4. Sistemul fuzzy folosit într-o casă inteligentă cuprinde două controlere fuzzy așezate în cascadă.
   Adevărat / Fals.

# 6

# Rețele neuronale artificiale

Prezentul capitol este o introducere în tema rețelelor neuronale artificiale prin descrierea istoriei și a tipurilor de rețele neuronale artificiale.

## 6.1. Introducere

Rețelele neuronale artificiale sunt tehnici de inteligență artificială care funcționează într-un mod asemănător sistemului nervos uman. În consecință, aceste tehnici de IA au capacitatea de a învăța. Asemenea unui nețele neuronale naturale, o reţea neuronală artificială (RNA) reprezintă un ansamblu de procesoare (neuroni) puternic interconectate, organizate în structuri masiv paralele şi care pot rezolva probleme complexe prin cooperarea între elementele simple de calcul.

Toate rețelele neuronale artificiale au abilități specifice lor: generalizarea, învățarea, sintetizarea, adaptabilitatea, aproximarea, robustețea, procesarea paralelă etc. Aceste abilități le fac foarte atractive în rezolvarea multor probleme, astfel că RNA împreună cu algoritmii de învățare automată, în prezent reprezintă cele mai utilizate tehnici de IA. În continuare se descrie succint aceste abilități ale RNA.

- Capacitatea de a învăța

  RNA nu necesită programe puternice ci sunt mai degrabă rezultatul unor antrenamente asupra unui set de date. Dându-se un set de intrări (eventual şi cu ieşirile dorite) în urma procesului de antrenament, reţelele neuronale se auto-organizează astfel că pot rezolva problemele pentru care au fost proiectate. Până în prezent, a fost dezvoltată o gamă largă de algoritmi de antrenament, fiecare având avantajele şi dezavantajele sale, putându-se aplica cu succes în unele cazuri sau fără nici o şansă de reuşită în altele.

- Capacitatea de a generaliza

Dacă au fost antrenate corespunzător, reţelele sunt capabile să dea răspunsuri corecte şi pentru intrări diferite faţă de cele cu care au fost antrenate, atât timp cât aceste intrări nu sunt foarte diferite. Este foarte important de menţionat că RNA generalizează automat ca urmare a structurii lor, nu cu ajutorul inteligenţei omului înglobată într-un anumit program special creat în acest scop.

- Capacitatea de a sintetiza

  RNA pot lua decizii sau trage concluzii când sunt confruntate cu informaţii complexe sau cu zgomote irelevante sau parţiale. De exemplu, o reţea poate fi antrenată cu o secvenţă de versiuni distorsionate ale literei A. Iar, după un proces de antrenare adecvat, aplicarea unei versiuni distorsionate a literei A va avea ca rezultat la ieşirea reţelei litera corectă. Deci, se poate spune că rețeaua a învăţat să producă ceva ce nu a mai văzut înainte.

- Capacitatea de a aproxima

  Rețelele neuronale artificiale pot aproxima orice funcție continuă, dacă au suficienți neuroni în stratul ascuns. Mai mult, chiar și o rețea cu un singur strat, care are o funcție de activare neliniară a neuronilor poate să aproximeze o funcție continuă dacă îndeplinește condiția amintită anterior. Această abilitate face din RNA o abordare flexibilă la probleme precum regresii, clasificare și controlul proceselor.

- Adaptabilitatea și plasticitatea

  Rețelele neuronale artificiale își modifică continuu ponderile interne în concordanță cu datele de intrare și factori externi, în timpul procesului de antrenare. Inspirată de la neuroni biologici, ponderile RNA sunt adaptate conform unor reguli bine definite.

- Robustețe și toleranța la defecțiuni

  RNA sunt reziliente la zgomot, date incomplete sau corupte. Datorită arhitecturii distribuite și numărului mare de unități de procesare, chiar dacă unii neuroni sau conexiuni funcționează defectuos, rețeaua va funcționa eficient și în aceste cazuri. Astfel, comparativ cu algoritmi tradiționali care dau erori în cazul unor date de intrare incomplete, RNA interpolează informațiile lipsă și va oferi predicții rezonabile.

- Procesare paralelă și distribuită

  Rețelele neuronale artificiale realizează calculele în paralel, ceea ce înseamnă că mai mulți neuroni vor procesa datele în același timp. Fiecare unitate de procesare a rețelei, neuronul funcționează individual, iar calculele au loc simultan de-a lungul rețelei. Datorită numărului mare de neuroni, la

apariția unui defect la un neuron, acest fapt nu va afecta eficacitatea rețelei, întrucât nu toată informația este concentrată într-un singur nod.

- Extragerea trăsăturilor și învățarea ierarhică
  Rețelele neuronale artificiale învață caracteristicile unor date, care le extrage din date brute neordonate. Mai mult, rețelele adânci formează o reprezentare ierarhică a datelor, întrucât straturile de început extrag caracteristicile, iar cele finale realizează abstractizarea conceptelor. Această abilitate a RNA le ajută să descopere modele ascunse ale bazelor de date de dimensiuni mari, ceea ce la face ideale pentru procesarea de semnale și detecția defectelor.
- Procesare secvențială și temporală
  Unele rețele neuronale artificiale precum cele recurente pot să își amintească date de intrare trecute și procesarea unor date secvențiale. Această capabilitate este esențială pentru prognoza datelor de tipul de serii de timp.
- Non-liniaritatea
  Contrar metodelor clasice, precum regresia, RNA pot modela relații neliniare. Funcțiile de activare introduc neliniaritatea, ceea ce dau posibilitatea RNA să rezolve probleme complexe de clasificare și regresie. Astfel, RNA sunt foarte efective în recunoașterea modelelor.

O sinteză a celor prezentate mai sus este descrisă în tabelul 6.1.

**Tabelul 6.1.** Ablitățile rețelelor neuronale artificiale

| **Abilitate** | **Descriere** | **Utilitate** |
|---|---|---|
| Generalizare | Aplică cele învățate asupra unor date neînvățate | Previne supraadaptarea și asigură adaptabilitatea la situații noi |
| Sintetizare | Extrage caracteristici de nivel înalt și reprezentări semnificative din date | Reduce dimensionalitatea și îmbunătățește procesul decizional |
| Învățarea | Determină cât de multă informație poate absorbi și reține un ANN. | Afectează complexitatea problemei pe care o poate gestiona rețeaua. |
| Aproximarea | Poate modela orice funcție | Rezolvă probleme complexe de clasificare |
| Adaptabilitate | Se adaptează la date noi prin actualizarea ponderilor. | Învață dinamic din experiență și din medii în schimbare. |

| | | |
|---|---|---|
| Robustețe | Gestionează datele zgomotoase, lipsă sau corupte. | ANN-urile sunt fiabile în aplicațiile din lumea reală. |
| Procesare paralelă | Procesează mai multe date simultan folosind calculul distribuit. | Permite o învățare rapidă și eficientă, în special cu GPU/TPU. |
| Extragerea trăsăturilor | Învață modele relevante automat din datele brute. | Elimină manipularea manuală a datelor și îmbunătățește acuratețea. |
| Memorie temporală | Memorează date de intrare trecute prin intermediul rețelelor recurente etc. | Ideal pentru operații secvențiale, cum ar fi prognozarea seriilor temporale. |
| Nonlinearitate | Modelează relații foarte neliniare prin funcțiile de activare. | Rezolvă limite de decizie complexe pe care modelele liniare nu le pot. |

Forma generală grafică a unei rețele neuronale artificiale este ilustrată în fig. 6.1.

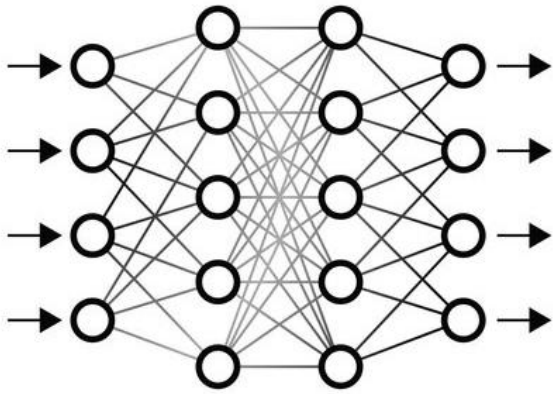


**Fig. 6.1.** Arhitectura generală a unei RNA

### 6.2. Istoria rețelelor neuronale artificiale

Istoria rețelelor neuronale artificiale a început atunci când oamenii de știință au reușit să înțeleagă funcționarea sistemului nervos uman, să modeleze matematic acest aspect, iar știința calculatoarelor și dispozitivele adiacente au ajuns la un nivel tehnologic suficient de ridicat încât să facă posibilă

implementarea acestor modele matematice, pentru a realiza un bagaj mare de calcule matematice.

Calculul neuronal se consideră că a debutat în anul 1943, când a apărut primul model de neuron artificial, în urma colaborării dintre un neurofiziolog și un matematician (Warren McCulloch şi Walter Pitts). Acest model este în linii generale acceptat şi în prezent. În fig. 6.2. este ilustrată analogia dintre neuronul biologic și neuronul artificial. Astfel, neuronul artificial are corpul celular asemănător neuronului biologic, primește datele de intrare de la alți neuroni sau exterior (la fel ca dendritele neuronului biologic), și transmite informația de ieșire către exterior sau către alți neuroni (prin intermediul axonului). În corpul neuronului sunt cuprinse relațiile matematice prin care se însumează datele de intrare, respectiv se folosesc funcții de activare pentru a obține informația de ieșire a neuronului. Neuronul artificial avea o formă binară (valoarea de ieșire a neuronului era 0 sau 1) și folosea o funcție logică simplă. Acest model a demonstrat faptul că o rețea de neuroni poate realiza orice operație logică simplă.

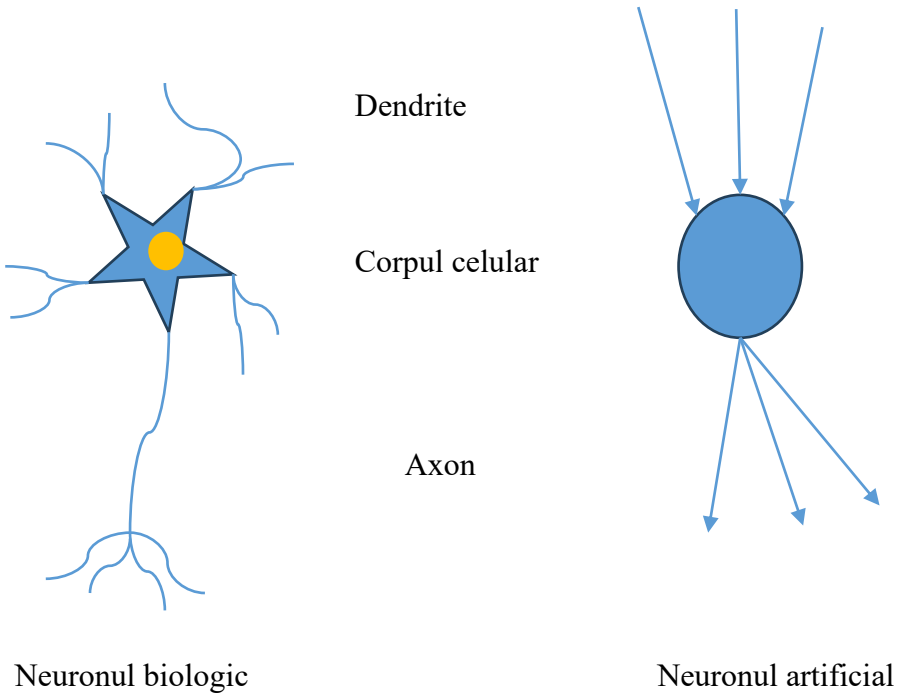



**Fig. 6.2.** Analogia neuronul biologic și neuronul artificial

Progresele înregistrate în neurobiologie şi psihologie au determinat apariția unor modele matematice ale învățării. Un model de învățare a fost propus în 1949

de către D.O.Hebb. Principiul introdus de Hebb este sintetizat des prin expresia ”neuronii care se declanșează împreună, se conectează împreună”. Această expresie se explică astfel: dacă doi neuroni se activează simultan, atunci legătura dintre ei se întărește, iar dacă doi neuroni nu se activează simultan, atunci conexiunea dintre ei slăbește. Modelul Hebb a reprezentat punctul de plecare în tentativele întreprinse în deceniul următor de realizare a unor rețele neuronale artificiale capabile să învețe.

În 1958 Frank Rosenblatt a publicat prima sa lucrare despre *perceptron*, prima rețea neuronală artificială capabilă să învețe. Despre acesta se specifica următoarele: „un model probabilistic pentru memorarea şi organizarea informaţiei în creier”. Modelul introdus de Rosenblatt putea clasifica modele separabile liniar folosind o regulă simplă de învățare, dar era limitat doar la acest tip de probleme.

ADALINE (Adaptive Linear Neuron) a fost introdus în 1959, la scurt timp după perceptronul lui Rosenblatt, de Bernard Widrow și Ted Hoff (unul dintre inventatorii microprocesorului) la Stanford. Ei au decis să construiască un mic dispozitiv electronic capabil să fie antrenat de algoritmul ADALINE pentru a învăța să clasifice modelele datelor de intrare. Principala diferență dintre perceptron și ADALINE este că cel din urmă funcționează prin minimizarea erorii pătratice medii a predicțiilor unei funcții liniare. Aceasta înseamnă că procedura de învățare se bazează pe rezultatul unei funcții liniare, iar în cazul perceptonului, rezultatul se bazează pe o funcție de prag. Diferența dintre modul de funcționare al unui perceptron și ADALINE este explicată în amănunt în [38].

Diferența dintre perceptron și ADALINE este procedura de învățare pentru a actualiza ponderile rețelei. Perceptronul actualizează ponderile calculând diferența dintre valorile datelor dorite și cele prezise / calculate. Cu alte cuvinte, perceptronul compară întotdeauna +1 sau -1 (valori prezise) cu +1 sau -1 (valori dorite). O consecință importantă a acestui fapt este că perceptronul învață doar atunci când se comit erori. În schimb, ADALINE calculează diferența dintre valoarea dorită a clasei y (+1 sau -1) și valoarea de ieșire continuă y^ din funcția liniară, care poate fi orice număr real. Acest lucru este crucial deoarece înseamnă că ADALINE poate învăța chiar și atunci când nu a fost făcută nicio greșeală de clasificare. Aceasta este o consecință a faptului că valorile de clasă prezise y′ nu influențează calculul erorii. Deoarece ADALINE învață tot timpul și perceptronul numai după erori, ADALINE va găsi o soluție mai rapid decât perceptronul pentru aceeași problemă [38].

Reţelele realizate în această perioadă au fost aplicate pentru rezolvarea unor probleme cum ar fi recunoaşterea unor structuri specifice din medicină (electrocardiograme) sau percepţia artificială. Astăzi, aceste tipuri de RNA sunt aplicaţii folosite pentru recunoaşterea formelor.

Perioada aniilor 1960-1970 este considerată iarna IA, din cauza unei lucrări publicate de Marvin Minsky și Seymour Papert. Într-adevăr, M. Minsky, unul dintre pionierii cercetători privind reţelele neuronale a întreprins o analiză lucidă a posibilităţilor şi limitelor modelelor neuronale existente în epocă. El a demonstrat imposibilitatea principală a reţelelor neuronale cu un singur strat de a rezolva probleme relativ simple. De exemplu, funcţia logică, SAU EXCLUSIV (XOR) nu putea fi realizată (calculată) de o astfel de reţea. Cartea lui M. Minsky şi P. Papert (Perceptrons, 1969), prin concluziile sale sceptice privind posibilitatea reţelelor multistrat de a depăşi dificultăţile perceptronului păstrând însă caracteristicile care-l fac interesant (liniaritatea, convergenţa procedurii de instruire, simplitatea conceptuală ca model de calcul paralel), a marcat o scădere dramatică a interesului pentru această direcţie de cercetare. Această lucrare prin care s-au subliniat limitările RNA primordiale a cauzat scăderea investițiilor și interesul în RNA.

Ieșirea din perioada de repaus a RNA a fost apariția în 1986 a cărţii „Parallel Distributed Proccesing, Explorations in the Microstructure of Cognition", de D. Rumelhart, J. McClelland şi grupul PDP. Acesta a fost considerat principalul eveniment care a marcat relansarea cercetărilor privind modelele conexioniste. În această lucrare, autorii au introdus algoritmul de învățare bazat pe propagarea înapoi a erorii (back-propagation). Prin acest algoritm se puteau antrena eficient rețele neuronale de tipul multistrat perceptron (multi-layer perceptron - MLP). La final, cu ajutorul acestei RNA și noul algoritm de învățare a fost posibilă rezolvarea problemei XOR, fapt care a reînviat interesul pentru această ramură a IA, ceea ce a dus la organizarea în 1987 a primei conferinţe internaţionale având ca principal actor reţelele neuronale artificiale.

Înainte de evenimentul amintit anterior, în anii 1982 și 1985 au avut loc trei evenimente: (1) John Hopfield a introdus rețelele neuronale recurente, (2) Teuvo Kohonen a propus algoritmul pentru învățarea nesupravegheată și (3) Hinton și Sejnowski au dezvoltat rețelele neuronale stohastice. Astfel, în 1982 John Hopfield a introdus un nou model de rețele neuronale artificiale, și anume memoria asociativă sau rețeaua Hopfield, care astăzi este clasificată ca un tip de rețea neuronală artificială recurentă. Prin această rețea, Hopfield a arătat cum se pot memora și extrage modelele / tiparele (patterns) dintr-o bază de date

neclasificată. Această rețea împreună cu învățarea Hebbiană au dus la nașterea unei RNA care stă la baza RNA moderne.

Tot în anul 1982, Teuvo Kohonen a dezvoltat un algoritm pentru învățarea nesupravegheată folosită în problemele de recunoaștere a tiparelor. Această metodă de învățare nesupravegheată este considerată cea mai apropiată metodă de învățare de modul în care învață oamenii, fără a avea un tipar de organizare a informațiilor. Rețeaua asociată acestui tip de algoritm de antrenare a fost denumit hartă auto-organizată (self-organizing maps – SOM) sau rețea Kohonen.

În 1985 sunt dezvoltate rețelele neuronale stohastice de către Hinton și Sejnowski, care au fost denumite mașini Boltzmann (Boltzmann Machines). Aceste rețele au capacitatea de a învăța probabilitatea distribuțiilor. Mașinile Boltzmann au fost unul dintre modelele inițiale care poate învăța reprezentarea internă a intrărilor. Mașina Boltzmann a fost creată pentru a rezolva două probleme, una era problema de căutare și alta era problema de învățare. Mașina Boltzmann conține neuroni vizibili și neuroni ascunși. Acest model a reprezentat una dintre cărămizile de bază a arhitecturilor de învățare adâncă (deep learning).

Ultima perioadă, cea a marilor descoperi în RNA a început în anii 2000 și se continuă și în prezent. Astfel, în 2006 Hinton și Salakhutdinov au introdus rețelele adânci, denumite de ei ”deep belief networks”, și au demonstrat că acestea pot fi învățate eficient strat cu strat. Șase ani mai târziu, în 2012, o rețea neuronală adâncă convoluțională (deep convolutional neural network) a câștigat competiția ImageNet competition, evidențiind astfel capabilitatea rețelelor adânci. IA modernă se consideră că a apărut în 2017, când Vaswani și colegii lui au dezvoltat rețele transformator (transformer networks), care au revoluționat soluționarea problemei de procesare a limbajului natural – NLP (Natural Language Processing). Într-adevăr, modele precum GTP, BERT și Vision Transformers (ViTs – transformatoare vizuale) au depășit, și înlocuit arhitecturile vechi de RNA. Mai mult, calculul neuromorfic și rețele neuronale de intensificare explorează modele inspirate din biologie.

O sinteză a istoriei rețelelor neuronale artificiale este ilustrată în fig. 6.4, în care se pot observa etapele principale în evoluția RNA, de la neuronul artificial, la prima rețea neuronală, la rețele transformator.

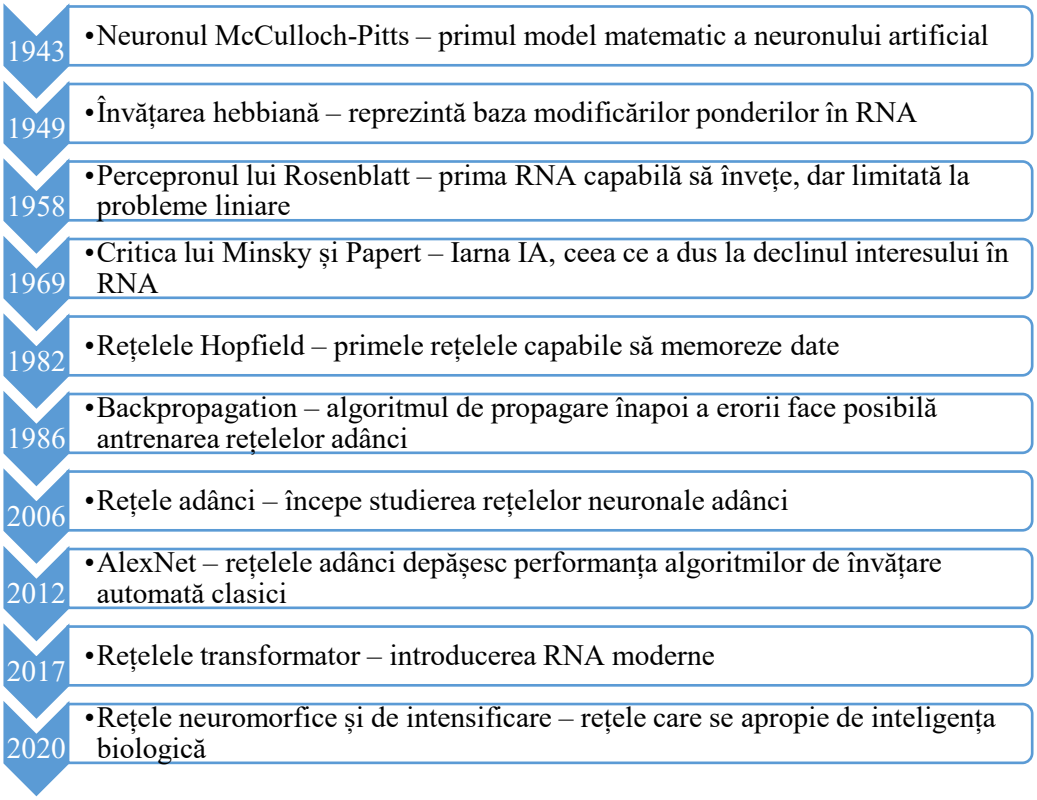



**Fig. 6.4.** Evoluția RNA

### 6.3. Tipuri de rețele neuronale artificiale

Rețelele neuronale artificiale pot fi clasificate în multe categorii în funcție de structura, arhitectura, numărul de neuroni, tipul neuronilor, sensul de propagare a informației, tipul de învățare, tipul funcției de activare a neuronilor, principiul de operare etc. În continuare, se vor prezenta toate tipurile de rețele neuronale artificiale.

În funcție de structura lor, RNA pot fi micro, meso sau macro rețele. Rețelele neuronale micro sunt RNA de dimensiuni mici cu o arhitectură simplă. Un exemplu de acest tip de rețele sunt rețelele perceptron cu un singur sau puține straturi, care se caracterizează printr-un număr mic de straturi și sunt folosite pentru rezolvarea problemelor simple de clasificare și regresie. Rețelele meso sunt rețele care au mai multe straturi sau conțin subrețele modulare. În această categorie intră rețelele convoluționale sau transformatoarele de dimensiuni medii. În final, rețelele macro se caracterizează printr-un număr foarte mare de neuroni (milioane sau miliarde), cu o structură complexă, care de cele mai multe ori cuprinde mai multe RNA modulare meso. Exemplele cele mai fidele din

această categorie sunt RNA transformator de tipul GPT și rețelele adânci cu învățare întărită (deep reinforcement learning systems).

În funcție de dimensiunea rețelei, aceasta poate fi pauci sau pluri. Astfel, RNA pauci au un număr mic de neuroni și straturi. Aceste rețele au un număr puțin de ponderi, astfel pot fi folosite pentru sarcini ușoare și nu au nevoie de învațare adâncă. Rețelele pluri sunt acele RNA care au un număr mare de neuroni și straturi. Aceste RNA au o arhitectură adâncă, fiind capabile să realizeze sarcini complexe și să manipuleze baze de date de dimensiuni mari. În această categorie intră transformatori, rețelele convoluționale adânci și și rețelele mari recurente. O comparație între cele două rețele arată că, din punct de vedere funcţional reţelele pauci-neuronale au un comportament strict deterministic, fiind din acest motiv utilizate în procesele logice simple, în timp ce reţelele pluri-neuronale manifestă un comportament probabilistic, fiind implicate în cele mai complexe procese nervoase

În clasificarea RNA după topologie, interesează modul în care neuronii sunt conectați între ei și cum informația circulă în interiorul rețelei. După acest criteriu sunt rețele feedforward, rețele recurente, rețele convoluționale, rețele modulare, rețele competitive și rețele hibride. Rețelele de tipul feedforward au drept caracteristică principală deplasare informației într-o singură direcție, de la stratul de intrare prin staturile ascunse la stratul de ieșire. În această categorie intră rețelele de tipul perceptron cu un singur strat de neuroni, multistrat perceptron și rețelele radiale. Rețelele recurente conțin neuroni care sunt interconectați prin conexiuni speciale, astfel încât informația circulă atât dispre intrare spre ieșire, cât și invers. În această categorie intră rețelele recurente simple, memoriile long short-term, unitatea recurentă închisă și rețelele de stare eco. RNA convoluționale folosesc convoluții (întoarceri) și acumularea pentru analiza datelor stațiale. RNA convoluționale tipice sunt considerate RNA convoluționale standard și complete și RNA capsule. Rețelele neuronale modulare cuprind mai multe rețele de mici dimensiuni interconecatate, care lucrează împreună. În această categorie, a RNA modulare, intră rețelele ierarhice, rețelele amsestec de experți (Mixture of experts) și comitet machines. Aceste tipuri de rețele au capacitatea să rezolve mai multe sarcini mici simultan prin intermediul RNA modulare interconectate. Rețelele competitive se caracterizează prin faptul că neuronii concurează între ei pentru a se activa, ci nu cooperează ca în cazul celorlalte tipuri de RNA. Aceste rețele se auto-organizează fără a avea nevoie de modele externe de învățare. Rețelele Hopfield, rețelele Kohonen și rețelele – teoria rezonanței adaptive intră în această categorie, a RNA competitive. În final,

sunt RNA hibride, care se caracterizează print-o performanță crescută obținută prin combinarea diferitelor tipuri de RNA. Acestea sunt rețelele neuro-fuzzy, rețelele adânci cu învățare întărită și transformatori. Fig. 6.4. ilustrează trei tipuri de RNA, și anume rețea feedforward (A), competitivă (B) și recurentă (C).

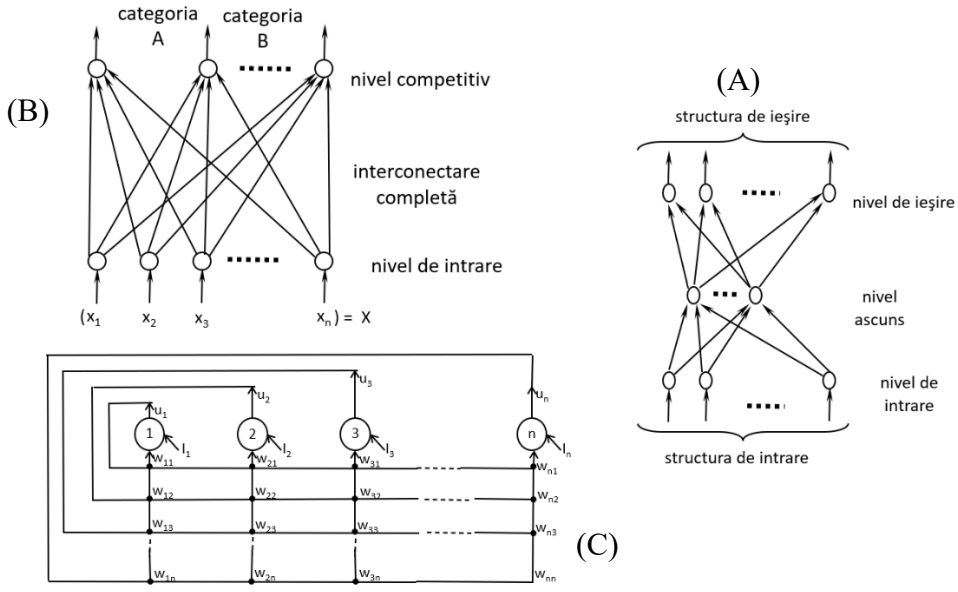



**Fig. 6.4.** Tipuri de RNA. (A) feedforward, (B) competitivă, (C) recurentă

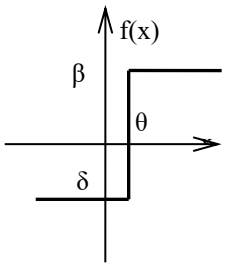
O altă abordare în clasificarea RNA este prin considerarea tipului de învățare folosită pentru antrenarea rețelei. În procesul de antrenare a RNA se pot folosi trei abordări, și anume învățarea supravegheată (supervizată - supervised), învățarea nesupravegheată (nesupervizată - unsuprevized) și învățarea prin întărire (reinforcement). Învățarea supravegheată este folosită atunci când sunt disponibile multe date de intrare și ieșirile corespunzătoare lor, prin intermediul lor RNA învață să asocieze datele de intrare la ieșirile corespunzătoare. Rețelele tipice antrenate prin învățare supravegheată sunt rețelele multi-strat perceptron și rețelele convoluționale. Învățarea nesupravegheată este utilizată în antrenarea rețelelor, care învață să recunoască tipare / modele în date de intrare neclasificate. Acest tip de abordare este aplicată în antrenarea rețelelor Kohonen și altor rețele competitive. Învățarea prin întărire operează prin acordarea de recompense și metodologia încearcă și eroare (trial-and-error). Ea se folosește în antrenarea rețelelor adânci și a celor asemănătoare lor.

Rețelele neuronale artificiale pot fi clasificate și dacă se ia în considerare funcționalitatea lor. Astfel, sunt rețele :

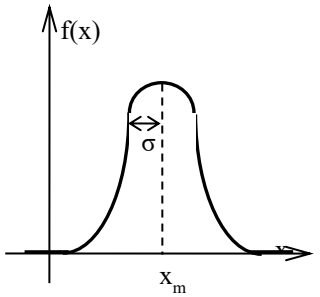
- Codificatoare automate (Autoencoders) – folosite pentru compresia datelor și învățarea caracteristicilor.
- Rețele generative (Generative Adversarial Networks - GAN) – care generează noi date, utilizate în tehnologia deepfake.
- Memoria Long Short-Term (LSTM) – un tip de rețele recurente care rezolvă problemele de gradient.
- Transformatori – folosite în procesarea limbajului natural (NLP), alimentează modele precum ChatGPT.

Datele cu care lucrează RNA pot să fie reprezentate prin diferite abordări, deci, în funcție reprezentarea datelor și a modului cum procesate datele, RNA sunt simbolice, subsimbolice și hibride. Modul cum funcționează fiecare dintre aceste rețele este: RNA simbolice procesează date discrete și logice, RNA subsimbolice lucrează cu date continue precum rețelele adânci tradiționale, iar cele hibride abordează procesarea simbolică și subsimbolică, deci pot procesa date complexe.

După modul de activare a neuronilor și se transferă semnalele dintre neuroni, RNA sunt rețele bazate pe prag, rețele sigmoidale, rețele rectificate și rețele de intensificare. Rețelele bazate pe prag utilizează pentru activarea neuronilor funcții prag precum modelul neuronului introdus de McCulloch și Pitts. Funcțiile sigmoid sau tangentă hiperbolică sunt folosite pentru o activare mai lină în cadrul rețelelor multistrat perceptron. Rețelele rectificate sunt variantele moderne ale RNA adânci, care folosesc funcția ReLU pentru activarea neuronilor. Cele mai complexe RNA sunt rețelele de intensificare care activează neuronii asemănător cu modul în care sunt activați neuronii biologici, adică folosirea unor vîrfuri discrete. Fig. 6.5. prezintă două funcții de activare folosite în RNA.



Funcţia de activare treaptă



Funcţia de activare gaussiană

**Fig. 6.5.** Funcții de activare

O altă abordare în clasificarea RNA este prin luarea în considerare a adaptabilității și capacitatea de evoluție a rețelelor. Din aceste puncte de vedere, RNA sunt rețele cu ponderile fixe, întrucât după realizarea antrenării, valorile ponderilor vor fi neschimbate pe toată durata funcționării rețelei. Alt tip sunt RNA cu ponderi adaptive. În acest cazul acestor rețele, ponderile se modifică continuu în timpul procesului de învățare. Cele mai avansate RNA sunt rețelele evolutive (neuroevolution), care folosesc algoritmi evolutivi pentru a modifica ponderile și structura rețelei.

În funcție de modelul computațional folosit, RNA sunt deterministe, probabilistice și cuantic. La RNA deterministe, ieșirile sunt fixe pentru aceleași intrări. Comparativ cu acestea, RNA probabilistice foosesc elemente stohastice pentru modelarea incertitudinilor, exemplu fiind rețelele Bayesiane. Rețelele neuronale cuantic explorează sisteme cuantic pentru RNA.

Dacă se clasifică RNA luând în considerare asemănarea lor cu sistemele biologice nervoase, acestea pot fi superficiale, adânci și neuromorfice. Primele RNA copiază doar principiile de bază care țin de structura și funcționarea sistemelor biologice. Următoarele RNA sunt inspirate de structurile ierarhice caracteristice sistemului nervos central uman. În final, ultimele RNA mimează activitatea neuronală reală prin implementarea fizică a rețelelor.

Și în sfârșit, RNA pot fi clasificate în două mari categorii în funcție de performanța lor în RNA cu performanță scăzută (RNA pauci), respectiv ridicată (GPT, AlphaGo).

O sinteză a tipurilor de RNA prezentate anterior și a metodelor de clasificare sunt ilustrate în fig. 6.6.

Din prezentarea tuturor tipurilor de RNA se poate observa imposibilitatea prezentării tuturor în amănunt, astfel în capitolele anterioare se vor descrie cele mai populare și utilizate tipuri de RNA și neuroni artificiali folosiți în rezolvarea problemelor din sistemelor electroenergetice și mai ales din managementul energiei. Astfel, în domeniul sistemelor electroenergetice au fost rezolvate probleme prin implicarea rețelelor feedforward, recurente, convoluționale, Kohonen, generative, adânci și de intensificare. Particularizând pentru subdomeniul de managementul energiei, RNA care sunt folosite pentru a rezolva probleme de optimizare, eficiență energetică și menținerea stabilității sunt rețelele feedforward, recurente, convoluționale, adânci, rețele antrenate prin întărire și desigur rețele hibride.

În capitolele următoare vor fi prezentate rețelele de tip feedforward, recurente, convoluționale, adânci și antrenate prin întărire.


**Fig. 6.6.** Clasificarea și tipuri de RNA

## 6.4. Întrebări și exerciții

1. Rețelele neuronale artificiale sunt singurele tehnici de IA care au capabilitatea de a învăța.
   Adevărat / Fals

2. Cele mai avansate RNA sunt transformatorii și rețelele neuromorfice.
   Adevărat / Fals

3. Rețelele cu învățare supravegheată pot fi folosite în orice situație.
   Adevărat / Fals

4. Rețelele adânci au înlocuit rețelele neuronale tradiționale.
   Adevărat / Fals

5. Prima rețea care a fost capabilă să rezolve toate operațiile logice a fost multistrat perceptron antrenat prin algoritmul de propagare înapoi a erorii.
   Adevărat / Fals.

# 7

# Rețele neuronale artificiale
## Arhitecturi fundamentale

Prezentul capitol descrie în detaliu rețelele neuronale artificiale care sunt caracterizate prin arhitecturi fundamentale, și care stau la baza tuturor rețelelor neuronale artificiale. Acestea sunt rețelele de tip feedforward, rețelele adânci și rețelele convoluționale.

### 7.1.Aspecte generale

Rețelele neuronale artificiale cu arhitecturi fundamentale stau la baza tuturor RNA moderne cu arhitecturi avansate. Acestea au în comun faptul că informația din interiorul RNA se deplasează într-o singură direcție, și anume dinspre stratul de intrare înspre stratul de ieșire, fără cicluri repetitive sau întoarceri. Deci, datele circulă unidirecțional prin aceste RNA. De asemenea, toate aceste rețele conțin un strat de intrare, unul sau mai multe straturi ascunse de neuroni și un strat de ieșire, iar informația trece prin RNA de la un strat la altul într-un mod secvențial.

Din punct de vedere compozițional, straturile RNA cuprind neuroni (denumire preluată din neurologie), dar în literatura de specialitate apar și sub numele de noduri sau unități / unități de procesare. Neuronii primesc datele de intrare de la alți neuroni sau din afara rețelei, pe care le procesează folosind ponderi și apoi le transmit mai departe prin funcțiile de activare asociate lor. Funcțiile de activare a neuronilor cele mai folosite sunt funcția sigmoidală, tangentă hiperbolică, ReLU sau softmax.

Neuronii care alcătuiesc RNA au diferite modele matematice. Câteva dintre modelele matematice sunt modelul static, modelul dinamic, modelul perceptron, modelul Kohonen și modelul cu vârfuri (spiking, cel mai apropiat de neuronul biologic).

Caracteristic RNA de bază este faptul că antrenarea rețelelor se realizează prin învățare supravegheată. În această situație este nevoie de seturi de antrenare, care conțin datele de intrare și ieșirile corespunzătoare (perechi intrare-ieșire), după care sunt ajustate valorile ponderilor prin abordarea diferitor metode de adaptare precum propagarea înapoi a erorii (back-propagation) și minimizarea gradientului (gradient descent). Deci, toate rețelele învață prin adaptarea ponderilor pe baza diferenței dintre mărimea de ieșire calculată și cea dorită (din setul de antrenament). Procesul de învățare a RNA este de multe ori optimizat prin folosirea unor tehnici de optimizare, precum optimizatorul Adam, pentru minimizarea erorii și îmbunătățirea predicților.

Diferența dintre aceste trei RNA constă în:

- Adâncimea rețelei – rețelele clasice feedforward au un singur strat ascuns, comparativ cu rețelele adânci care au cel puțin trei straturi ascunse. Rețelele convoluționale sunt rețele adînci, dar dedicate pentru date spațiale, deci straturile sunt ierahizate.
- Arhitectura – rețelele feedforward au straturile complet interconectate, pe când cele convoluționale au straturile parțial conectate.
- Aplicații – rețelele feedforward sunt folosite pentru rezolvarea de rețele simple precum clasificare și aproximarea funcțiilor. Rețelele adânci sunt folosite în recunoașterea vorbirii, limbajului natural și imaginilor. Iar, rețelele convoluționale sunt implicate în procesarea imaginilor și video-urilor, precum și clasificarea imaginilor și a detecției de obiecte.

O sinteză a caracteristicilor celor trei arhitecturi fundamentale ale RNA este prezentată în tabelul 7.1.

**Tabelul 7.1.** Caracteriticile comune ale RNA fundamentale

| **Caracteristică** | **Feedforward** | **Adânci** | **Convoluționale** |
|---|---|---|---|
| Circulația informației | Intrare - > ieșire | Intrare - > ieșire | Intrare - > ieșire |
| Învățarea | Supravegheată | Supravegheată | Supravegheată |
| Adaptare | Backpropagation Gradient descent | Backpropagation | Backpropagation |
| Arhitectură | Simplă – 1, 2 straturi ascunse | Adâncă – minim 3 straturi ascunse | Adâncă – straturi convoluționale |

| | | | |
|---|---|---|---|
| Utilizare | Clasificare simplă | Recunoaștere de tipare complexe | Procesare de imagini / video |
| Neuroni | Modelul static | Modelul static, dinamic | Modelul static, dinamic |

### 7.2.Rețelele feedforward

O rețea neuronală artificială feedforward, RNAF este un tip de RNA în care datele circulă într-o singură direcție, dinspre intrare, înspre ieșire. Structura de bază a unei RNAF de tip multistrat perceptron este ilustrată în fig. 7.1. Se poate observa în figură stratul de intrare la nivelul 1. Unităţile nivelului de intrare sunt singurele care primesc semnale externe rețelei. Unităţile din nivelul intermediar (stratul ascuns) sunt complet interconectate cu unităţile din nivelurile de intrare şi de ieşire. De asemenea, nu există conexiuni între unităţile din acelaşi nivel. Numărul de niveluri intermediare (ascunse) poate fi mai mare decât unu, dar dacă este mai mare sau egal cu trei atunci rețeaua intră în categoria RNA adânci. Ultimul nivel, nivelul 3 este nivelul de ieșire, neuronii lui fiind cei care transmit rezultatul final.

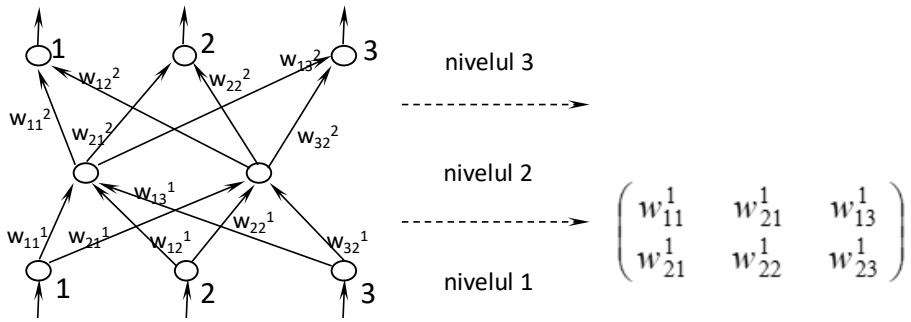



**Fig. 7.1.** Structura unei RNAF

Ponderile asociate fiecărei conexiuni sinaptice sunt ajustate în timpul procesului de antrenare (învăţare) a reţelei. La finalul acestui proces, ponderile rămân fixate atât pe durata fazei de testare cât şi în timpul utilizării reţelei în rezolvarea problema date.

Fiecare neuron al rețelei (exceptând cei din stratul de intrare) procesează datele care ajung la el prin folosirea ponderilor și a funcției sale de activare, rezultatul transmițându-l la alți neuroni (sau ca ieșire în cazul neuronilor din

stratul de ieșire). În continuare sunt descrise modelele matematice ale neuronilor artificiali folosiți în cadrul RNAF.

Neuronul dinamic este cea mai completă reprezentare a neuronului ca element de bază al unei RNA. Așa cum arată fig. 7.2., el conţine trei componente: un sumator ponderat, un sistem liniar dinamic şi o funcţie neliniară de ieşire.

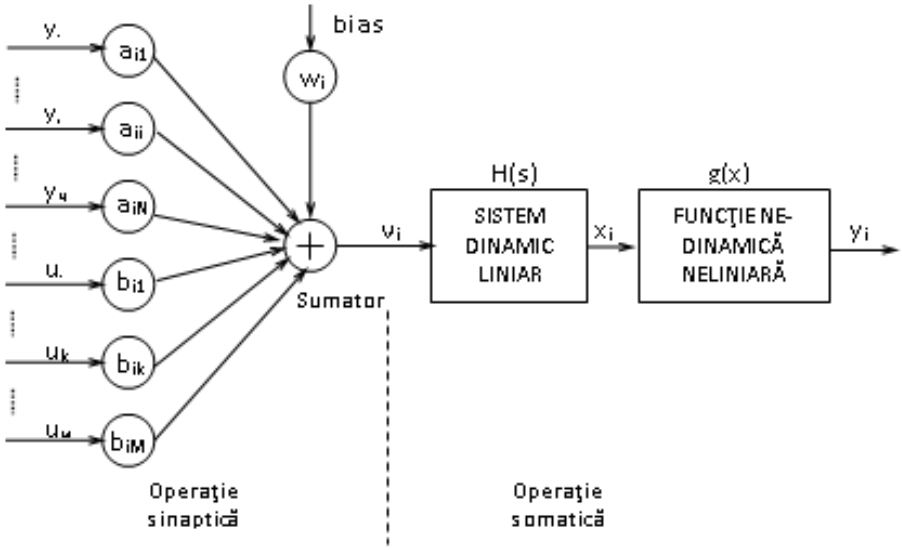



**Fig. 7.2.** Structura neuronului dinamic

Elementele componente ale neuronului au următoarele funcții:

- Sumatorul ponderat operează conform ecuaţiei

$$v_i(t) = \sum_{j=1}^{N} a_{ij} y_j(t) + \sum_{k=1}^{M} b_{ik} u_k(t) + w_i \qquad (7.1)$$

$y_i$ sunt intrările provenite de la alţi neuroni, $u_k$ sunt intrările simple din exterior, $a_{ij}$, $b_{ik}$ sunt ponderile intrărilor, iar $w_i$ este o constantă denumită „bias”.

- Sistemul liniar dinamic este de tip SISO (Single Input Single Output) şi are rolul de a imprima neuronului o anumită comportare dinamică. Sistemul este descris de o funcţie de transfer *H(s)*, cu echivalentul *h(t)* în timp. Cinci variante de sistem dinamic sunt descrise în tabelul 7.2.
- Funcţia neliniară de ieşire (sau de activare) $y_i = g(x_i)$ are rolul de a da o anumită formă ieşirii neuronului. Ea poate fi diferenţială (continuă) sau

nediferenţială. De asemenea, funcţia poate avea formă de impuls sau de treaptă şi poate avea valoarea medie nulă (ieşire bipolară) sau pozitivă (ieşire monopolară). Tabelul 7.3. prezintă cinci funcţii neliniare de ieşire.

**Tabelul 7.2.** Sisteme dinamice a neuronului dinamic

| Domeniul Laplace | Domeniul timp | Relaţia intrare-ieşire a SISO |
|---|---|---|
| $H(s)=1$ | $h(t)=\delta(t)$ | $x_i(t)=v_i(t)$ |
| $H(s)=\frac{1}{s}$ | $h(t)=\begin{cases}0, & t<0\\1, & t\geq 0\end{cases}$ | $\dot{x}_i=v_i(t)$ |
| $H(s)=\frac{1}{1+sT}$ | $h(t)=\frac{1}{T}e^{\frac{-t}{T}}$ | $T\dot{x}_i(t)+x_i(t)=v_i(t)$ |
| $H(s)=\frac{1}{\alpha_0 s+\alpha_1}$ | $h(t)=\frac{1}{\alpha_0}e^{-\frac{\alpha_1}{\alpha_0}t}$ | $\alpha_0\dot{x}_i(t)+\alpha_1 x_i(t)=v_i(t)$ |
| $H(s)=e^{-sT}$ | $h(t)=\delta(t-T)$ | $x_i(t)=v_i(t-T)$ |

unde $\delta(t)$ este funcţia Delta Dirac.

**Tabelul 7.3.** Funcţii neliniare de ieşire a neuronului dinamic

| **Numele funcţiei** | **Formula** | **Caracteristici** |
|---|---|---|
| Prag monopolar | $g(x)=\begin{cases}+1, x>0\\0, \rightleftarrows\rightleftarrows\rightleftarrows altfel\end{cases}$ | Nediferenţiabilă, treaptă |
| Prag bipolar | $g(x)=\begin{cases}+1, x>0\\-1, \rightleftarrows\rightleftarrows\rightleftarrows altfel\end{cases}$ | Nediferenţiabilă, treaptă |
| Sigmoidă | $g(x)=\frac{1}{1+e^{-x}}$ | Diferenţiabilă, treaptă, medie pozitivă |
| Tangentă hiperbolică | $g(x)=tanh(x)=\frac{1-e^{-2x}}{1+e^{-2x}}$ | Diferenţiabilă, treaptă, medie nulă |
| Gaussian | $g(x)=e^{-x^2-a^2}$ | Diferenţiabilă, impuls, medie pozitivă |

În general, este dificil de a se stabili care dintre funcţiile de ieşire este mai potrivită pentru o aplicaţie dată. Astfel, există tipuri de RNA în care anumite funcţii de ieşire dau rezultate mai bune decât altele. Într-adevăr, neuronii biologici localizaţi în diferite părţi ale sistemului nervos au caracteristici de ieşire diferite. În literatura de specialitate au fost propuse şi alte funcţii de ieşire: logaritmice, exponenţiale etc.

Neuronul static se obţine din neuronul dinamic prin omiterea funcţiei liniare dinamice. În consecință, un neuron static este un caz particular, simplificat al celui dinamic pentru care $H(s) = 1$, adică $x_i(t)=v_i(t)$. În cazul RNAF, acest neuron este unitate de procesare tipică.

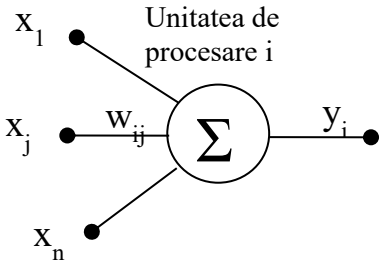



**Fig. 7.3.** Structura neuronului static

Se observă în fig. 7.3. că, fiecare intrare are asociată o pondere a conexiunii, exprimată ca $w_{ij}$ unde $i$ este unitatea destinatară, iar $j$ este unitatea emiţătoare. La fel ca la neuronul real, unitatea de procesare are mai multe intrări şi o singură ieşire, care se poate răspândi la multe alte unităţi din reţea. Semnalul de intrare primit de unitatea $i$ de la neuronul $j$ este notat cu $x_j$.Această valoare reprezintă de asemenea ieşirea nodului $j$, la fel cum ieşirea unităţii $i$ este etichetată $y_i$. Există analogii între ponderile conexiunilor neuronilor artificiali şi modelul neuronului biologic:

- ieşirile unităţii de procesare corespunde frecvenţei de inervare a neuronului;
- ponderea conexiunii corespunde forţei de conexiune sinaptică între neuroni.

Valorile ponderilor sunt numere reale, dar de cele mai multe ori se alege să se folosească numere subunitare pozitive, adică intervalul [0,1]. În alte situații, ele pot fi pozitive (semnificând conexiuni excitatorii între neuroni) sau negative (semnificând conexiuni inhibitorii).

Unitatea de procesare individuală realizează câteva operaţii foarte simple. Deci, fiecare unitate determină o valoare netă de intrare, pe baza tuturor conexiunilor sale de intrare. În absenţa unor conexiuni speciale, în mod tipic valoarea netă se calculează însumând valorile de intrare înmulţite cu ponderile corespunzătoare.

$$net_{y_i} = \sum_{j=1}^{n} w_{ij} x_j \tag{7.2}$$

unde *j* parcurge toate conexiunile de intrare ale unităţii *i*.

Se evidențiază astfel că, excitarea şi inhibarea neuronilor sunt luate în considerare automat, prin semnalul ponderilor lor. Acest calcul al sumei de produse joacă un rol important în simularea reţelei. Datorită faptului că puterea reţelei stă în numărul mare de conexiuni, viteza cu care poate fi calculată valoarea $net_i$ pentru fiecare unitate din reţea, determină performanţele reţelei simulate.

Odată ce intrarea netă este calculată, ea este convertită într-o valoare de activare a unităţii de procesare prin folosirea relației:

$$a_i(t) = F_i[a_i(t-1), net_{y_i}(t)]. \tag{7.3}$$

Activarea curentă a neuronului poate depinde de valoarea precedentă a activării. Această dependenţa a fost inclusă în definiţie pentru generalitate. Pentru cazul unei funcţii de activare de tip identitate se poate scrie

$$a_i = net_{y_i} \tag{7.4}$$

care corespunde în mod uzual modelului neuronului static

Se defineşte valoarea de ieşire a unităţii, prin aplicarea unei funcţii de ieşire $y_i = f_i(a_i) = f_i(net_{y_i})$, unde $f_i$ este funcţia de ieşire sau funcţia prag.

În analiza şi proiectarea multor modele de RNA este utilă formularea vectorială a relaţiilor (7.2) – (7.4). Dacă se consideră o reţea neuronală compusă din mai multe niveluri de unităţi de procesare identice și se presupune că un anumit nivel conţine *n* unităţi, atunci ieşirile acestui nivel pot fi văzute ca un vector *n* – dimensional.

$$X = [x_1, \ldots, x_n]^t \tag{7.5}$$

Se presupune că acest vector de ieşire *n* – dimensional furnizează valorile de intrare pentru fiecare unitate aparţinând unui nivel *m* – dimensional superior.

Fiecare unitate a nivelului *m* – dimensional va avea *n* ponderi asociate cu conexiunile din nivelul anterior. În consecinţă, există *m* vectori pondere *n* – dimensionali asociaţi cu acest nivel. Vectorul pondere corespunzător unităţii *i* poate fi scris sub formă matriceală în mod similar.

$$W_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^t \tag{7.6}$$

Intrarea netă în unitatea *i* poate fi exprimată ca produsul intern al vectorului de intrare şi al vectorului pondere:

$$net_{y_i} = \sum_{j=1}^{n} x_j w_{ij} = X^t \cdot W_i \tag{7.7}$$

$$net_{y_i} = X^t \cdot W_i; X^t \cdot W_i = W_i^t \cdot X \tag{7.8}$$

Relaţiile vectoriale de acest tip sunt utile în abordările teoretice ulterioare.

Neuronul formal, denumit și neuronul McCulloch-Pitts, poate fi modelat ca un dispozitiv cu mai multe intrări, neliniar, cu conexiuni ponderate (ponderi) $w_{ij}$, fig. 7.4. Varianta electronică a acestui neuron este ilustrată în fig. 7.5.

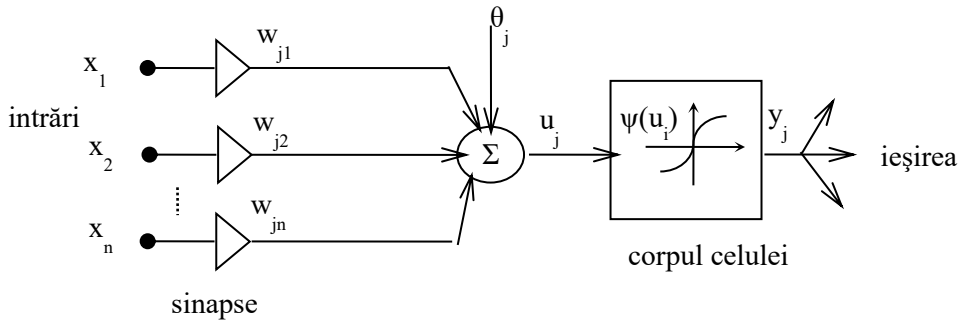



**Fig. 7.4.** Structura neuronului formal

Corpul celulei este reprezentat de o funcţie prag neliniară, $\Psi(u_j)$. Cel mai simplu model al unui neuron artificial sumează cele *n* intrări ponderate şi pasează rezultatul printr-o neliniaritate la ieşire, conform ecuaţiei.

$$y_i = \Psi\left[\sum_{i=1}^{n} w_{ji} x_i + \theta_j\right] \tag{7.9}$$

unde $\Psi$ este o funcţie de limitare sau prag, numită şi funcţie de activare sau funcţie de transfer neliniară, $\theta_j$ este un prag extern numit offset sau bias corespunzător unităţii bias fictive, $w_{ij}$ sunt ponderile sinaptice ale conexiunilor, $x_i$ sunt intrările unităţii de procesare, $n$ este numărul de intrări ale unităţii de procesare $y_j$ este ieşirea unităţii de procesare.

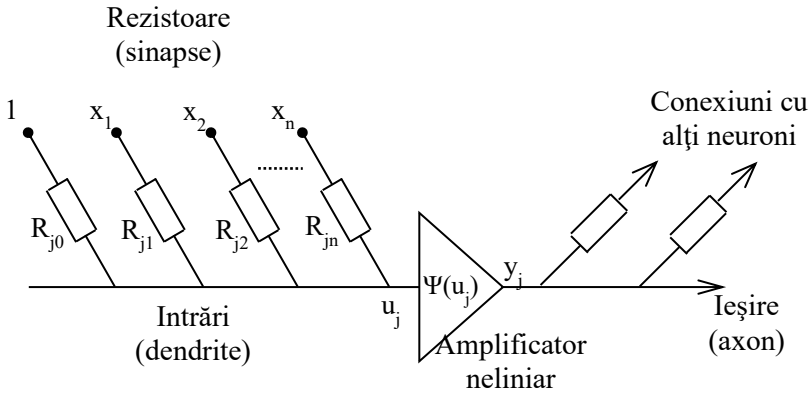



**Fig. 7.5.** Structura neuronului formal – model electric

Neuronul artificial formal este caracterizat de neliniaritatea sa şi de offsetul $\theta_j$. Modelul McCulloch-Pitts al neuronului foloseşte ca funcţie de activare funcţia prag binară, fig. 7.6.

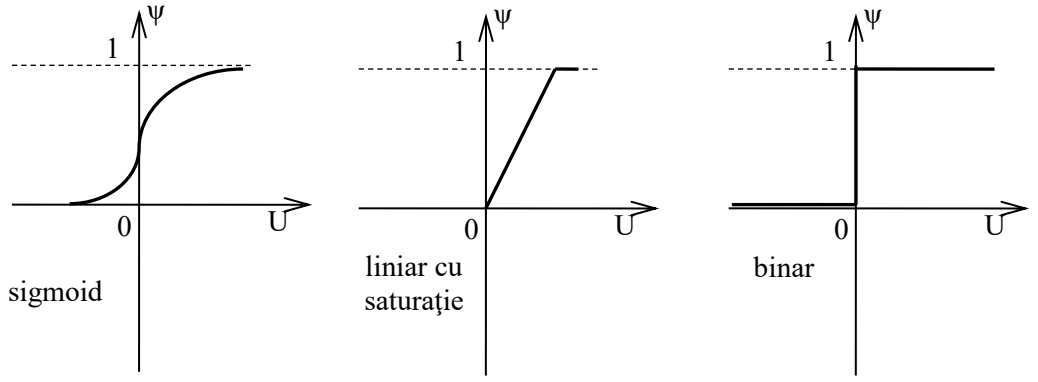



**Fig. 7.6.** Funcţii de activare a neuronului formal

Funcţia prag poate fi înlocuită de o funcţie neliniară mai generală şi în consecinţă, ieşirea neuronului poate lua valori într-un set discret {-1,1} sau {0,1}

sau poate varia continuu, de obicei între două valori limită $y_{min}$ şi $y_{max}$, cu $y_{max} > y_{min}$.

Nivelul de activare sau starea neuronului este măsurat(ă) prin valoarea semnalului de ieşire, care este în mod uzual determinat de o funcţie monoton crescătoare (sigmoid) a sumei ponderate a semnalelor de intrare. Această funcţie este descrisă de relaţiile

$$u_j = \sum_{i=0}^{n} w_{ji} x_i \tag{7.10}$$

$$y_j = \tanh(\gamma \cdot u_j) = \frac{1 - e^{-2\gamma u_j}}{1 + e^{-2\gamma u_j}} \tag{7.11}$$

$\gamma$ este un număr real pozitiv, constant sau variabil, care controlează panta funcţiei sigmoid

În comparaţie cu funcţia prag liniar, funcţia sigmoid este mult mai adecvată pentru implementările hard analogice; de asemenea este mai realistă din punct de vedere al neuronilor biologici.

Neuronul probabilistic diferă de modelele tradiționale prezentate anterior prin încorporarea incertitudinii în activarea neuronilor și actualizările ponderilor. Astfel, în loc de o funcție deterministă (exemplu - ReLU Sigmoid), care este folosită pentru activarea neuronilor statici, dinamici și formali, un neuron probabilistic este activat pe baza unei distribuții de probabilitate. Acest model este util atunci când problema de rezolvat se ocupă de date cu multe zgomote, incerte sau incomplete, ceea ce face acest model neuronal extrem de relevant pentru rezolvarea problemelor din sistemele electroenergetică în condiții de incertitudine.

Neuronul probabilistic are aceeași structură ca neuronul static, dar activarea se realizează printr-o funcție de forma

$$P(y = 1|x) = \sigma(Wx + b) \tag{7.12}$$

unde $\sigma(x)$ este funcția sigmoid, prin care ieșirile sunt în intervalul (0, 1); $Wx + b$ este suma ponderilor cu intrările.

Neuronul va fi activat atunci când probabilitatea este $P(y = 1)$ în loc să aibă valori de 0 sau 1. Deci, valoarea de ieșire a neuronului este deci o probabilitate

În ceea ce privește valorile ponderilor, care dacă în cazurile celorlalte modele au valori fixe, în cazul modelelor probabilistice, ponderile se modifică după o

distribuție probabilistică (cea mai populară este distribuția Gaussiană). Acest aspect dă capacitatea unei RNA care folosește neuroni probabilistici să estimeze încrederea (certitudinea) în predicțiile rezultate.

Tot un neuron probabilistic este neuronul Bayesian, care are ca trăsătură definitorie ponderi caracterizate de incertitudine, adică distribuția probabilistică a ponderilor.

Avantajele modelelor probabilistice sunt capacitatea de a manipula incertitudini, de a estima încrederea predicțiilor și de a se adapta probabilitățile în timp real pentru sistemele dinamice. Contrar, punctele slabe ale neuronilor probabilistici sunt calculele mai complexe care durează mai mult timp, dificultatea în antrenarea acestora și utilizarea lor în situații în care datele sunt deterministe, ceea ce face ca modelele probabilistice să nu fie necesare.

Neuronul stohastic introduce aleatorietatea în procesul de activare, ceea ce face ca acest model nruonal să fie mult mai robust decât precedentele. Astfel, în locul unei funcții de activare care determină funcționarea neuroilor, neuronii stocastici sunt activați pe baza unui prag de probabilitate; deci probabilitatea de activare a neuronilor depinde de distribuția probabilității de activare.

Neuronii stocastici apar sub forma a trei modele: modelul binar, modelul Gaussian și modelul Poisson. Structural, acești neuroni au aceleași componente ca neuronul static, adică intrări, ponderi, corp neuronal. Diferența constă în modul în care se obține ieșirea neuronului și valoarea acesteia. Activarea se realizează conform relației următoare.

$$P(y|x) = f(\sum w_i x_i + b) \tag{7.13}$$

Apoi ieșirea *y* este eșanționată pe baza acestei probabilități.

Neuronul stohastic binar are două posibile valori de ieșire: 1 sau 0. Astfel, ieșirea are forma, și este egală cu 1 în cazul relație (7.14), contrar are valoarea 0.

$$P(y = 1) = \sigma(\sum w_i x_i + b) \tag{7.14}$$

În cazul neuronului stohastic Gaussian, ieșirea se determină prin eșationarea mulțimii $N(\mu, \sigma^2)$, unde $\mu$ este suma ponderată.

Neuronii stocastici Poisson, care sunt folosiți în calculul neuronal neuromorfic, modelează comportamentul spiking prin activarea neuronilor pe baza procesului Poisson

$$P(activat\ în\ \Delta t) = \lambda \Delta t \tag{7.15}$$

Unde $\lambda$ este rata de activare a neuronului.

În multe situații, în procesul de activare a neuronilor stocastici se introduce un zgomot

$$P(y|x) = f(\sum w_i x_i + b + \eta) \tag{7.16}$$

unde $\eta$ este un zgomot de tip Gaussian sau Poisson, și influențează starea finală a neuronului.

Deoarece, antrenarea acestor neuroni presupune lucrul cu incertitudini, algoritmul de propagare înapoi a erorii trebuie modificat. În consecință sunt folosite abordări bazate pe eșationare (divergența contrastantă) sau reparametrizări (transformare diferențială a zgomotului).

Neuronul fuzzy este un model neuronal care integrează aspecte de logica fuzzy, întrucât acest neuron nu folosește valori singulare numerice, ci numere fuzzy. Datorită acestui aspect, neuronii fuzzy sunt folosiți în construcția de RNA care au capabilitatea de a rezolva probleme caracterizate de date imprecise, incomplete și incertitudini. Deoarece, în capitolul 14 se vor prezenta tehnici AI hibride, în acest capitol nu se va insista pe neuronul fuzzy, care este unul hibrid. Acesta, împreună cu RNA fuzzy vor fi descriși în amănunt îen capitolul menționat.

Neuronul cuantic este o unitate de calcul care integrează mecanica cuantică în modelele de rețele neuronale. Acești neuroni folosesc stările cuantice, suprapunerea, amestecarea și porțile cuantice pentru a efectua calcule, oferind avantaje potențiale în paralelism, eficiență și capacitatea de învățare față de neuronii clasici. Spre deosebire de neuronii clasici, care procesează valori numerice, neuronii cuantici procesează qubiți (biți cuantici) care pot exista în suprapunere (atât 0, cât și 1 simultan). Acști neuroni nu sunt încă implementați în structură mari, ci doar la scară mică, dar modele teoretice există și se lucrează cu ei.

Un aspect esențial în utilizarea RNA este antrenarea acestora printr-o abordare care presupune un set de date de antrenament, care conțin pecehi de date de intrare și ieșirea corespunzătoare; adică învățarea supravegheată (supervised learning). Pentru RNAF, așa cum s-a mai precizat, cea mai populară metodă de învățare este prin propagarea înapoi a erorii (Backpropagation error).

***Învățarea prin backpropagation*** presupune ca după prezentarea fiecărui exemplu de antrenare rețelei, ponderile sunt ajustate astfel încât diferența dintre ieşirea reţelei şi ieşirea dorită să fie minimizată. Trebuie menționat faptul că exemplele din setul de antrenare sunt prezentate reţelei de mai multe ori, iar procesul de prezentare a setului de antrenare o singură dată a reţelei este definită ca o epocă. Deci, o „epocă" conţine un număr de iteraţii egal cu numărul exemplelor de antrenare.

Procesul de antrenare se desfășoară după diagrama din fig. 7.7.

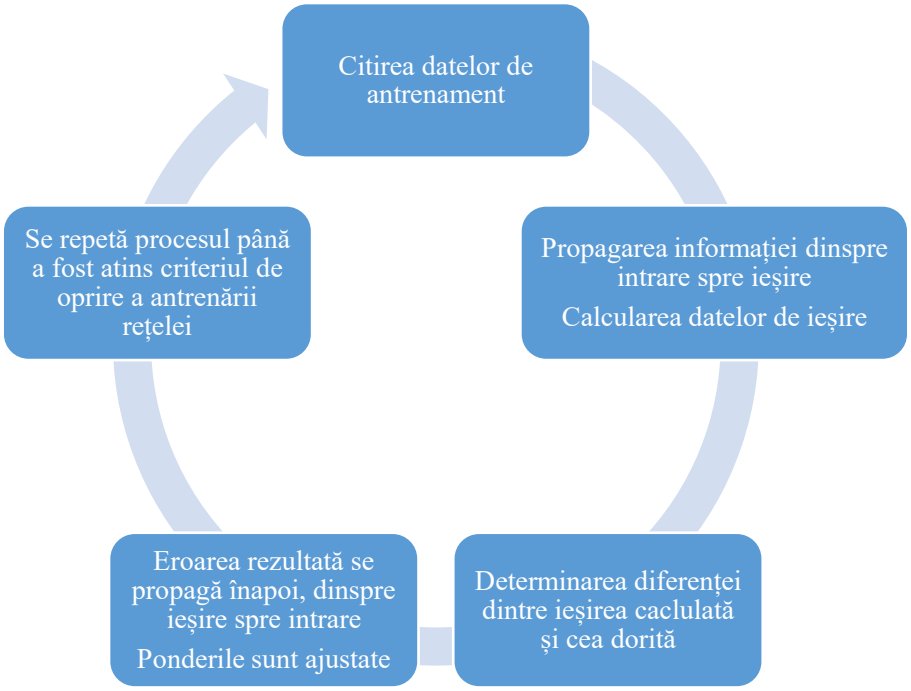



**Fig. 7.7.** Diagrama procesului de antrenare prin backpropagation

Se observă în figură faptul că, fiecare iteraţie a algoritmului de antrenare constă dintr-un pas de propagare înainte, urmat de un pas de propagare înapoi. Mai mult, se asociază fiecărei unităţi de procesare o valoare a erorii ($\delta$), care reflectă cantitatea de eroare asociată cu acea unitate. Parametrul $\delta$ este folosit în procedura de corecţie a ponderilor, în timpul procesului de învăţare. Dacă $\delta$ are o valoare mare, rezultă că trebuie aplicată o corecţie mare asupra ponderilor care vin spre acea unitate, iar semnul erorii $\delta$ determină direcţia în care trebuie modificate ponderile. Relațiile matematice corespunzătoare procesului de actualizare / ajustare a ponderilor în etapa de antrenare a RNA sunt (7.17) și (7.18).

$$\delta = y_D - y_c \tag{7.17}$$

$$w_{ij}^{*} = w_{ij} + \Delta w_{ij} \tag{7.18}$$

unde noua valoare adaptată a ponderii este $w_{ij}^{*}$, $\alpha$ este rata de învățare, care ia valori în intervalul [0, 1], $\Delta w_{ij} = \alpha \cdot x_j \cdot dw_{ij}$, $dw_{ij} = x_j(1 - x_j) \cdot \delta$; $\Delta w_{ij}$ este valoarea cu care se modifică ponderea, iar $dw_{ij}$ este ponderea din eroare care revine lui $w_{ij,}$

În concluzie, reţeaua învaţă un set predefinit de perechi de exemple de intrare – ieşire, folosind ciclul de două faze „propagare - adaptare" pentru fiecare exemplu. După ce a fost aplicată structura de intrare pentru primul nivel de unităţi din reţea, ea se propagă prin fiecare nivel superior până când este generată o ieşire. Structura de ieşire este apoi comparată cu structura dorită şi este calculat semnalul de eroare pentru fiecare unitate de ieşire. Semnalele de eroare sunt transmise înapoi de la nivelul de ieşire la fiecare nod intermediar care contribuie direct la ieşire. Fiecare unitate din nivelul intermediar primeşte doar o porţiune din semnalul total de eroare, pe baza contribuţiei relative adusă de unitatea respectivă la eroarea totală. Fiecare unitate îşi va reiniţializa ponderile, în funcţie de semnalul de eroare primit, pentru a determina reţeaua să conveargă spre o stare stabilă în care toate structurile de antrenament să fie învăţate. Ca urmare a acestui proces, în timpul antrenării reţelei, unităţile de procesare din nivelul intermediar se organizează ele însele, astfel încât noduri diferite învaţă să recunoască caracteristici diferite ale structurii totale de intrare. Dacă după antrenare, reţelei i se prezintă o structură de intrare cu zgomot sau incompletă, similară însă cu o structură pe care reţeaua a învăţat să o codifice în timpul antrenării, unităţile din nivelurile ascunse vor răspunde cu ieşiri active

Mai mult a fost demonstrat că reţeaua tinde să dezvolte relaţii interne între noduri astfel încât să organizeze datele de antrenament în clase de structuri. Această tendinţă poate fi extrapolată la ipoteza că toate unităţile din nivelurile ascunse în ale RNA sunt asociate într-un anume fel cu trăsături specifice ale structurii de intrare, ca rezultat al antrenării. În consecinţă, reţeaua va găsi la sfârşitul fazei de antrenare o reprezentare internă care-i permite să genereze ieşirea dorită pentru o structură de intrare dată. Astfel, dacă reţelei i se prezintă o structură de intrare pe care nu a mai văzut-o, reţeaua va încerca să o clasifice în concordanţă cu trăsăturile pe care le-a învăţat la antrenare.

***Învățarea prin minimizarea gradientului*** este un algoritm de optimizare utilizat pentru a minimiza eroarea (sau funcția de cost) a unui model prin ajustarea iterativă a parametrilor acestuia (ponderi și bias). Scopul este de a minimiza o funcție *f*(*w*), unde *w* reprezintă parametrii (ponderile). Algoritmul de minimizare a gradientului se actualizează în mod iterativ prin deplasarea în direcția opusă gradientului funcției. Noua valoare a unei ponderi, care trebuie ajustată este calculată cu relația (7.19). Procesul de adaptare continuă până când funcția converge spre o valoare minimă.

$$w^* = w - \alpha \nabla f(w) \tag{7.19}$$

unde $\alpha$ este rata de învățare, și controlează cât de mult se va modifica noua pondere, iar $\nabla f(w)$ este gradientul funcției (prima derivată) față de *w*.

Funcția *f* reprezintă funcția de cost / funcția de pierderi este etalonul după care se determină cât de departe este rezultatul calculat față de cel dorit (din setul de antrenament). În procesul de adaptare, valoarea acestei funcții se urmărește să se minimizeze spre o valoare dată. Funcția de cost se poate determina prin implicarea următoarelor: Mean Squared Error (MSE) – eroarea medie pătratică (7.20) și cross-entropy loss – pierderea de entropie încrucișată (7.22).

$$f(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{7.20}$$

unde *n* este numărul de eșantioane, $y_i$ este valoarea reală (dată prin setul de antrenare), iar $\hat{y}_i$ este valoarea prezisă (calculată).

Iar gradientul funcției este

$$\frac{df(w)}{dw} = \frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)(-x_i) \tag{7.21}$$

$$f(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i log\hat{y}_i + (1 - y_i) log(1 - \hat{y}_i))^2 \tag{7.22}$$

Iar gradientul funcției cross-entropy loss este

$$\frac{df(w)}{dw} = \frac{1}{n}\sum_{i=1}^{n}(\widehat{y_i} - y_i)x_i \tag{7.23}$$

Un ultim aspect despre funcția de cost – aceasta poate lua alte forme, dar condiția de bază este ca aceasta să fie derivabilă, pentru a se putea determina gradientul.

În literatura de specialitate sunt mai multe variante a acestui tip de învățare, prin care sunt adaptate toate ponderile, sau doar anumite ponderi.

***Învățarea prin distribuția probabilistică Gaussiană,*** care se folosește în cazul neuronilor probabilistici și stocastici, presupune modelarea incertitudinii în activările neuronilor, actualizarea ponderilor pe baza inferenței probabilistice și valorificarea proprietăților statistice ale distribuțiilor gaussiene pentru a îmbunătăți stabilitatea învățării și generalizarea. Mai exact:

- Pentru neuronii probabilistici, învățarea implică estimarea densităților de probabilitate condiționată de clasă folosind funcții gaussiene, așa cum se vede în rețelele neuronale probabilistice (PNN). Aceste rețele calculează probabilitatea ca o intrare să aparțină unei anumite clase folosind funcții gaussiene ale nucleului și aplică reguli de decizie bayesiene pentru a clasifica datele. Probabilitatea ca intrarea $x$ să aparțină clasei $c$ este

$$P(c|x) = \frac{\sum_{i \in C} e^{-\frac{\|x - x_i\|^2}{2\sigma^2}}}{\sum_{j=1}^{N} e^{-\frac{\|x - x_j\|^2}{2\sigma^2}}} \tag{7.24}$$

unde $x_i$ eșantionul de antrenament al clasei $c$, $\sigma^2$ parametru de netezire (lățimea nucleului), $N$ – numărul total de eșantioane, $C$ – set de monstre care aparține clasei $c$.

Clasa $c^*$ care are cea mai mare valoare a probabilității este selectată

$$c^* = \arg\max\left(P(c|x)\right) \tag{7.25}$$

- Pentru neuronii stocastici, învățarea implică introducerea aleatoriei în activări prin extragerea ieșirilor neuronilor dintr-o distribuție gaussiană. Acest lucru este folosit în mașinile Boltzmann, rețelele neuronale bayesiene și autoencoderele variaționale, unde zgomotul gaussian este încorporat în învățarea modelării incertitudinii și a preveni

supraadaptarea. Astfel, în neuronii stocastici, activările sunt eșantionate dintr-o distribuție Gaussiană în loc să fie deterministe:

$$a_i = F(\mu_i, \sigma_i^2) \tag{7.26}$$

unde $a_i$ activarea neuronului $i$, $F$ distribuția Gaussiană, $\mu_i$.partea deterministă a activării neuronului, $\sigma_i^2$ incertitudinea din activare (se modifică în timpul antrenării)

- În inițializarea ponderilor, distribuțiile gaussiene ajută la definirea greutăților inițiale în rețelele neuronale profunde, asigurând propagarea echilibrată a semnalului și prevenind gradienții de dispariție/explozie. Inițializarea corectă a ponderilor folosind distribuții gaussiene previne dispariția/explozia gradienților. Cele două inițializari principale sunt inițializarea Xavier (7.27) și inițializarea He (7.28). Aceste inițializări asigură că variația activărilor rămâne stabilă între straturi, prevenind saturația sau explozia.

$$w \sim F(0, \frac{1}{n_{in}}) \tag{7.27}$$

unde $n_{in}$ numărul de intrări a neuronului

$$w \sim F(0, \frac{2}{n_{in}}) \tag{7.28}$$

- În algoritmii de antrenament, algoritmul Expectation-Maximization (EM) este utilizat pentru a actualiza iterativ parametrii în modelele Gaussian Mixture (GMM), permițând învățarea nesupravegheată sau semi-supravegheată a distribuțiilor complexe.

Distribuția gaussiană și funcția $N(\mu, \sigma^2)$ (care reprezintă o distribuție normală) sunt direct legate. Notația $N(\mu, \sigma^2)$ este o modalitate compactă de a exprima faptul că o variabilă aleatoare urmează o distribuție gaussiană (normală) cu o medie dată $\mu$ și varianță $\sigma^2$. O variabilă X urmărește o distribuția Gaussiană, dacă respectă condiția:

$$P(X = x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{7.29}$$

Astfel, tehnicile de învățare bazate pe distribuția Gaussiană permit învățarea supraveheată cu incertitudini, clasificarea robustă și proprietăți de convergență îmbunătățite în rețelele neuronale.

Rețelele neuronale artificiale feedforward cuprind mai multe tipuri de RNA, care toate au proprietățile și modul de funcționare prezentate anterior. Aceste rețele sunt: perceptron cu un singur strat (SLP), perceptron multistrat (MLP), rețea cu funcție de bază radială (RBFN), rețeaua neuronală convoluțională (CNN), rețea neuronală profundă (DNN), mașină de învățare extremă (ELM), rețea neuronală cu întârziere (TDNN), rețea neuronală modulară (MNN), Ecou State Network (ESN), și rețea neuronală de regresie generalizată (GRNN).

### 7.3.Rețelele adânci

Rețelele neuronale adânci sunt o extensie a RNA feedforward de tip multistrat perceptron, care se caracterizează printr-un număr de straturi ascunse mai mare de 3 (în practică, numărul de straturi ascunse este 5, 10 sau mai mare). Structura unei rețele adânci este precum cea prezentată în fig. 7.1., cu excepția unui număr de cel puțin trei straturi ascunse (fig. 7.8). Rolul stratului de intrare este de a prelucra datele de intrare, stratul de ieșire produce datele de ieșire, iar straturile ascunse extrag caracteristicile datelor de intrare pe diferite niveluri de abstractizare. Astfel, primul strat de neuroni ascunși este pentru extragerea caracteristicilor și două sau mai multe straturi suplimentare pentru a permite ierarhii mai profunde de caracteristici și modele mai bune de tipare complexe.

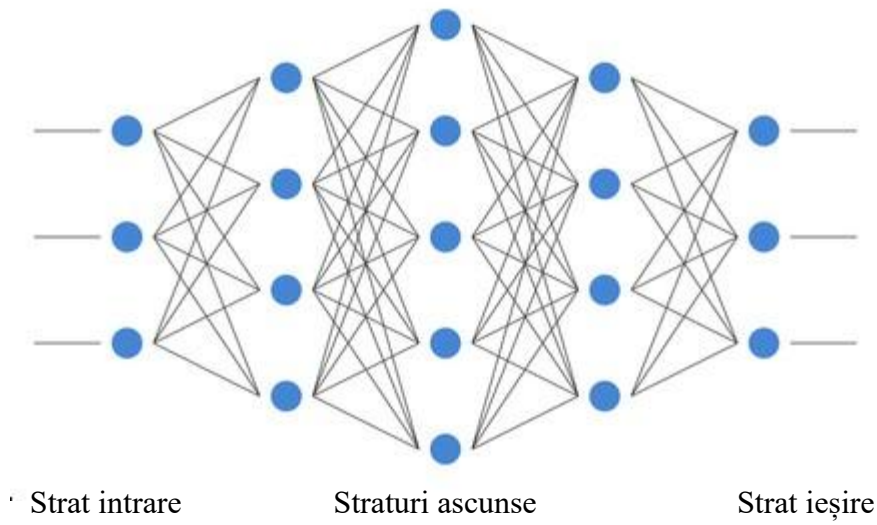



**Fig. 7.8.** Structura generală a unei RNA adânci

Avantajele rețelelor adânci față de cele clasice sunt:

- Învață mai bine tipare complexe (de exemplu, imagini, vorbire) – poate extrage caracteristici ierarhice.
- Îmbunătățește generalizarea dacă sunt instruite corespunzător – pentru a se realiza antrenarea corespunzătoare se folosesc algoritmi de optimizare precum Adam RMSprop.
- Reduce nevoia de manipulare manuală a caracteristicilor.
- Aproximează mai eficient funcțiile neliniare.

Principalul dezavantaj constă în faptul că necesită mai multe date și putere de calcul decât rețelele de mică adâncime.

O sinteză a comparației dintre rețelele feedforward clasice și cele adânci este prezentată în tabelul 7.4.

**Tabelul 7.4.** Comparație între rețelele adânci și cele clasice

| **Aspect** | **RNA clasice** | **RNA adânci** |
|---|---|---|
| Straturi ascunse | 1 | Cel puțin 3 |
| Extragerea trăsăturilor | Trăsături simple | Trăsături abstracte, ierarhizate |
| Capacitatea de învățare | Linitată la funcții simple | Poate modela funcții complexe, neliniare |
| Complexitatea antrenării | Ușoară, puține calcule | Dificilă, necesită efort computațional |
| Supra/sub-adaptarea | Predispuse la subadaptare | Predispuse la supraadaptare |
| Tehnici de optimizare | Algoritmii de bază | Necesită tehnici de optimizare |
| Manipularea datelor mari | Limitată | Se pot manipula date mari |
| Generalizarea | Limitată | Bună |

Exemple de RNA adânci include versiuni profunde ale altor rețele, plus rețele neuronale adânci cu feedforward (DFNN), rețele de credință profundă (DBN – deep belief networks), rețele transformatoare (de exemplu, BERT, GPT).

### 7.4.Rețelele convoluționale

Rețelele convoluționale, RNAC se diferențiază de alte RNA feedforward prin faptul că au fost dezvoltate cu scopul de a îndeplini anumite sacini specifice care presupun prelucrarea datelor spațiale sau a imaginilor.

Comparativ cu RNA descrise anterior, la care toate straturile de neuroni sunt la fel (de același tip), rețele convoluționale au trei tipuri de straturi: convoluționale, de punere în comun (pooling) și complet conectate.

Straturi convoluționale au rolul de a extrage caracteristicile locale prin conectarea la stratul de intrare sau precedent. Straturi de punere în comun se folosesc pentru reducerea dimensiunilor spațiale (down-eșantionare) pentru a face rețeaua mai eficientă și pentru a reduce cantitatea de calcule necesare. Straturi complet conectate sunt ultimele straturile prin care ieșirea este aplatizată și trecută prin aceste straturi dense (similar cu RNA feedforward clasică). Deci, rețelele convoluționale sunt concepute special pentru a exploata corelația spațială locală prezentă în imagini sau date similare.

Capabilitatea de a manipula date spațiale, ceea ce face RNAC ideale pentru lucru cu astfel de date provine de la straturile convoluționale prin care rețeaua păstrează ierarhia spațială a datelor de intrare, mai mult arhitectura rețelei o ajută să învețe automat ierarhi spațiale.

O caracteristică a RNAC este folosirea în comun a ponderilor, adică aceeași nucleu (filtru) este aplicat în diferite părți ale datelor de intrare. Datorită acestui fapt, numărul parametrilor este redus mult și modelul poate recunoaște tipare în datele de intrare indiferent de poziția lor în imagine.

RNAC prezintă câmpuri locale receptive la care sunt conectați neuronii din stratul de intrare sau precedent într-un mod local. Această conexiune locală ajută rețeaua să învețe trăsături de nivel redus precum margini sau colțuri, care apoi le combină pentru recunoașterea tiparelor evidențiate prin niveluri mai mari. Astfel, comparativ cu RNA clasice, aceste rețele au mai puțini parametrii, datorită câmpurilor locale și a ponderilor comune, deci sunt mai eficiente și mai puțin predispuse la supra-adaptare.

Extragerea automată a trăsăturilor de către RNAC se realizează în două etape de abstractizare:

- Etapa 1 – folosirea straturilor convoluționale de început pentru extragerea trăsăturilor de bază (tipare simple).
- Etapa 2 – straturile adânci extrag trăsăturile fine, caracteristice tiparelor complexe.

Antrenarea RNAC presupune folosirea algoritmului backpropagation, dar filtrele din straturile convoluționale sunt actualizate / adaptate prin operația de convoluție. Aceste filter (nuclee - kernels) sunt învățate în timpul procesului de antrenare a rețelei. În concluzie, RNAC sunt capabile să se focalizeze pe trăsăturile relevante ale datelor de intrare.

Tabelul 7.5. sintetizează diferențele și asemănările dintre RNA clasice și cele convoluționale în funcție de mai multe aspecte.

**Tabelul 7.5.** Comparație între rețelele adânci și cele clasice

| **Aspect** | **RNA clasice** | **RNA convoluționale** |
|---|---|---|
| Arhitectura | Complet conectată | Trei tipuri de straturi |
| Recunoaștere spațială | Nu | Da |
| Parametrii | Număr mare de parametri | Număr mai mic de parametri |
| Extragerea trăsăturilor | De suprafață sau manuală | Automată, |
| Câmp receptiv | Global | Local |
| Utilizare | Date tabelare, probleme simple | Clasificarea imaginilor, a datelor spațiale |
| Antrenare | Complexitate moderată | Mai complex |
| Tipul datelor | Date generale sub formă de vectori și matrice | Imagini, date spațiale |

Procesul de învățare în RNAC urmează aceleași principii fundamentale ca și în RNA feedforward, dar primele au mecanisme speciale care fac antrenarea lor mai eficientă și mai potrivită pentru datele spațiale.

Propagarea datelor dinspre intrare spre ieșire se realizează prin aplicarea relația (2.30), prin care se efectuează operația de convoluție la nivelul straturilor convoluționale. Activarea neuronilor din restul straturilor se realizează conform relațiilor matematice uzuale folosite de RNA feedforward clasice.

$$z_{i,j,k}^{(l)} = \sum_{m}\sum_{n}\sum_{c} K_{m,n,c,k}^{(l)} \cdot a_{i+m,j+n,c}^{(l-1)} + b_{k}^{(l)} \qquad (7.30)$$

unde $z_{i,j,k}^{(l)}$ este activarea neuronului de pe poziția ($i$, $j$) a nucleului $k$, din stratul $l$; $K_{m,n,c,k}^{(l)}$ – ponderile nucleului $k$;

$a_{i+m,j+n,c}^{(l-1)}$ – activarea din straturile precedente din locația ($i$+$m$, $j$+$n$) pe canalul $c$; $b_k^{(l)}$ – termenul bias a filtrului $k$.

Propagarea înapoi a erorii (backpropagation) în rețelele convoluționale este realizată ca gradient a operației de convoluție. Astfel, în cazul acestor rețele sunt ajustate nucleele, ci nu matricea de ponderi. Gradientul pentru ponderile nucleelor (7.31), respectiv a activării neuronilor de intrare (7.32) sunt descrise în următoarele paragrafe.

$$\frac{\partial L}{\partial K_{m,n,c,k}^{(l)}} = \sum_{i,j} \delta_{i,j,k}^{(l)} \cdot a_{i+m,j+n,c}^{(l-1)} \tag{7.31}$$

$$\frac{\partial L}{\partial a_{i,j,c}^{(l-1)}} = \sum_{m,n,k} K_{m,n,c,k}^{(l)} \cdot \delta_{i-m,j-n,c}^{(l-1)} + \tag{7.32}$$

Stratul de punere în comun (pooling) are rolul de a reduce dimensiunea calculelor. Ecuația de maxim a punerii în comun folosită de rețea este (7.33), iar gradienul ecuației este (7.34).

$$a_{i,j,k}^{(l)} = \frac{max}{(m,n) \in R(i,j)} a_{m,n,k}^{(l-1)} \tag{7.33}$$

unde $R(i, j)$ este regiunea de punere în comun (pooling) de dimensiune PxP.

$$\frac{\partial L}{\partial a_{m,n,k}^{(l-1)}} = \begin{cases} \frac{\partial L}{\partial a_{i,j,k}^{(l)}}, & dacă\ a_{m,n,k}^{(l-1)} = \max\ \mathrm{R}(i,j) \\ 0, & contrar \end{cases} \tag{7.34}$$

În concluzie, RNAC învață ierarhiile spațiale în timp ce reduc calculul prin folosirea în comun a ponderilor. Propagarea înapoi a erorii în aceste rețele implică gradienți convoluționali, ceea ce o face mai eficientă decât rețelele complet conectate. RNAC necesită mai puțină memorie și mai puțini parametri, făcându-le scalabile pentru date cu dimensiuni mari.

Rețelele neuronale convoluționale cuprinde următoarele tipuri de rețele - rețea neuronală convoluțională standard (CNN), rețea complet convoluțională

(FCN), rețea neuronală reziduală (ResNet), DenseNet (rețele convoluționale dens conectate), rețea capsule (CapsNet).

### 7.5.Întrebări și răspunsuri

1. Principalele arhitecturi de bază a RNA sunt
   Rețelele feedforward / Rețelele recurente / Rețelele simple

2. Învățarea supravegheată presupune folosirea unor perechi de antrenament intrare-ieșire corespunzătoare.
   Adevărat / Fals

3. Algoritmul de propagare înapoi a erorii este strategia de bază în antrenarea rețelelor de tipul feedforward.
   Adevărat / Fals

4. Cel mai folosit model neuronal este
   Neuronul static / Neuronul dinamic / Neuronul formal

5. Rețelele neuronale adânci sunt diferite de RNA clasice prin faptul că au două straturi de intrare.
   Adevărat / Fals

6. Rețelele neuronale convoluționale sunt dedicate pentru manipularea datelor spațiale.
   Adevărat / Fals

7. În rețelele neuronale artificiale feedforward datele circulă într-un singur sens.
   Adevărat / Fals.

8. Rețelele neuronale feedforward cuprind
   Rețele cu un singur strat / Rețele cu mai multe straturi / Rețele convoluționale / Rețele recurente

# 8

# Rețele neuronale artificiale
## Arhitecturi avansate

Prezentul capitol continuă tematica rețelelor neuronale artifiale prin focalizarea pe rețele mai complexe și dinamice, precum RNA recurente, de întărire, hibride și rețelele competitive. În consecință, sunt explorate arhitecturi avansate care introduc un parametru nou, timpul, și folosesc adaptarea și învățarea nesupravegheată.

### 8.1.Aspecte generale

Rețelele neuronale artificiale caracterizate de arhitecturi avansate cuprind unități de calcul, neuronii, care sunt interconectați între ei, dar față de cei descriși în capitolul precedent, au structuri mai complexe. Asemenea RNA de bază, RNA complexe sunt antrenate printr-un algoritm de antrenare, iar apoi testate și validate prin intermediul unor baza de date. Comparativ cu RNA fundamentale, rețelele cu arhitecturi avansate sunt mai diverse, ele fiind folosite în multe aplicații, însă toate pot fi caracterizate de trăsături comune, și anume: procesarea în funcție de timp, adaptabilitatea, bucle de feedback, generalizarea în diverse domenii, metode avansate de învățare și dinamica antrenării neliniare și complexe.

Natura dinamică

Natura dinamică a RNA avansate rezultă din două direcții: dependența de timp și evoluția bazată pe stare. Într-adevăr, multe dintre aceste rețele procesează date secvențiale în timp sau au un aspect temporal. De exemplu, rețelele recurente se ocupă în mod explicit de secvențe de date temporale, iar abordarea folosită în învățare se face prin consolidare (reinforcement learning), care se concentrează pe bucla de feedback în timp (state-action-reward / stare-acțiune-recompensă). Mai mult, rețelele recurente cât și cele antrenate prin consolidare, se bazează pe

modele în care starea evoluează în timp în funcție de intrările sau acțiunile anterioare.

Adaptabilitatea

Adaptabilitatea implică faptul că aceste RNA învață din experiență, astfel folosesc învățarea prin feedback sau din interacțiunea cu mediul. De exemplu, rețelele recurente mențin stări ascunse, care evoluează pe baza intrărilor anterioare, permițându-le să-și amintească informațiile din trecut. Mai mult, modelele de învățare prin consolidare (DQN, Policy Gradients) învață din recompense sau pedepse prin încercări și erori, adaptându-și politica pentru a maximiza recompensa. De asemenea, rețelele hibride se pot adapta pe baza arhitecturii lor.

Mecanisme de feedback / răspuns

Unele rețele precum cele recurente realizează un feedback intern, unde ieșirea unui nivel devine intrarea pentru următorul. Acest mecanism de feedback este esențial pentru învățarea secvenței sau reprezentările de stare în descrierea evoluție. Alte rețelele, precum cele care au la bază învățarea prin întărire implică și feedback, dar sub formă de recompense sau penalități primite în urma acțiunilor într-un mediu, care influențează deciziile (acțiunile) viitoare.

Generalizare pe domenii

Generalizarea pe domenii se referă la utilizarea multifuncțională a rețelelor. Astfel, rețelele recurente sunt folosite într-o gamă largă de sarcini secvențiale, precum prognoza seriilor de date temporale. Învățarea prin consolidare este utilizată în special în domenii care necesită luarea deciziilor în timp, cum sunt aplicațiile - sisteme autonome. Iar, rețelele hibride, cum ar fi codificatoarele automate (encoder-ele), sunt implicate în rezolvarea unor probleme care necesită sarcini generative și învățarea de reducere a dimensionalității/reprezentării.

Utilizarea tehnicilor avansate de învățare

Majoritatea RNA avansate utilizează o variantă avansată a adaptării ponderilor prin propagarea înapoi a erorii. De exemplu, rețelele recurente folosesc backpropagation în timp pentru a actualiza ponderile și a-și optimiza modelele. Arhitecturile hibride folosesc retropropagarea într-o configurație adversă specifică (antrenament generator și discriminator).

Procese complexe de antrenare

Antrenarea rețelelor recurente se confruntă adesea cu provocarea de a dispărea /exploda gradienții, iar tehnici precum tăierea gradientului sunt implicate pentru a stabiliza învățarea. De asemenea, rețelele care folosesc învățarea prin întărire pot avea un proces de învățare instabil cauzat de variația

recompenselor și a dilemei de explorare-exploatare. Alte rețele necesită un echilibru între antrenarea unității generatoare și cea discriminatoare pentru a preveni ca prima să o copleșească pe a doua.

Neliniaritatea

Toate RNA cu arhitecturi avansate folosesc funcții de activare neliniare (exemplu - sigmoidală, ReLU – unitate liniară rectificată, tangentă hiperbolică) pentru a introduce neliniaritatea în modelul neuronilor componenți, permițându-le să învețe tipare și reprezentări complexe.

Aceste similarități dintre RNA complexe sunt întărite de diferențele dintre ele, care sunt descrise în amănunt în restul capitolului, însă o sinteză a acestor diferențe este prezentată în tabelul 8.1.

**Tabelul 8.1.** Diferențele dintre RNA complexe studiate

| Aspect | Rețele neuronale artificiale | | | |
|---|---|---|---|---|
| | Recurente | Învățate prin întărire | Hibride | Competitive |
| Feedback | Da | Uneori | Variază | Nu |
| Învățarea | Supravegheată modificată | Prin întărire | Mixtă Supravegheată, nesupravegheată, competitivă | Nesupravegheată |
| Principalul scop | Memorie secvențială | Luarea deciziei în timp | Modelare complexă | Grupare date |
| Aplicații | Serii de timp, control | Control | Compresia datelor | Clasificare |

## 8.2.Rețele recurente

Rețelele neuronale artificiale recurente, RNAR conțin neuroni care sunt conectați între ei prin cicluri direcționate, astfel informația din rețea va persista în timp. Aceste rețele sunt dedicate prelucrării datelor secvențiale prin folosirea unor stări ascunse care crează o memorie internă în rețea.

Arhitectura standard a unei RNA recurente este ilustrată în fig. 8.1. Se observă în figură un strat de intrare, două straturi ascunse și un strat de ieșire. De asemenea, imaginea prezintă și ponderile dintre neuroni, care pot conecta toți

neuronii între ei sau doar parțial. O caracteristică definitorie este existența conexiunilor recurente, care leagă neuroni din straturi superioare de cei din straturi inferioare. Există RNA, care cuprind ponderi recurente care se buclează și în jurul unui neuron. Prin această buclă de informație se crează o memorie internă prin repetarea datelor.

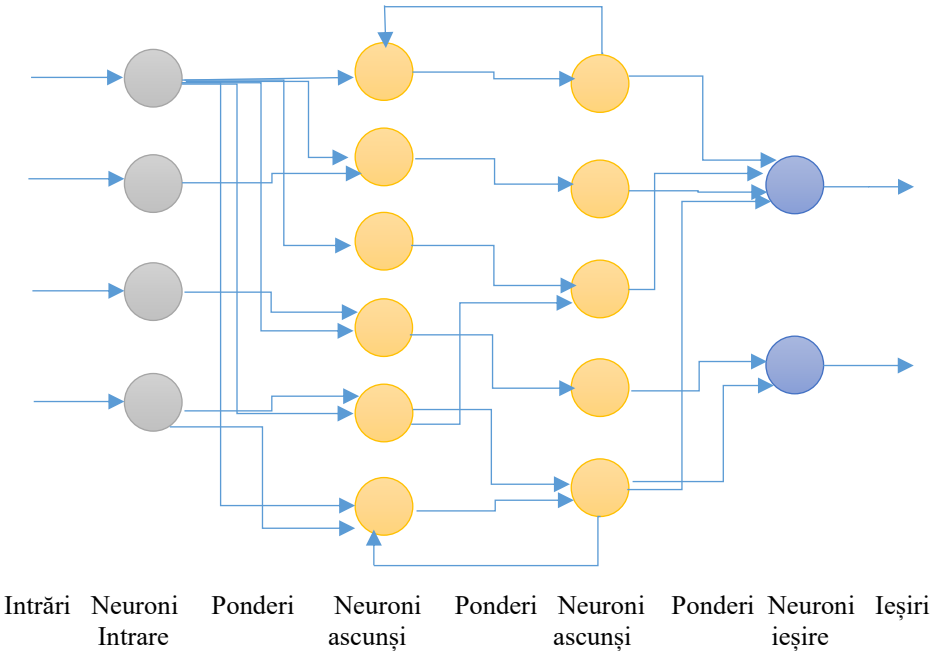



**Fig. 8.1.** Arhitectura unei rețele recurente standard

Modul de funcționare a unei rețele recurente este asemănător modului de funcționare a unei rețele feedforward; diferența constă în felul în care sunt tratate ponderile recurente și adaptarea algoritmului de învățare. Mai exact, fiecare moment $t$ este caracterizat de valori specifice ale ponderilor și neuronilor. Relațiile matematice (8.1) și (8.2) prezintă modul în care se determină activarea unui neuron din stratul ascuns, respectiv a unui neuron din stratul de ieșire.

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{8.1}$$

$$y_t = g(W_{hy}x_t + b_y) \tag{8.2}$$

unde $f$, $g$ sunt funcția de activare a neuronului din stratul ascuns (sigmoid sau tanh), respectiv a neuronului din stratul de ieșire (sigmoid sau softmax), $h_t$ este neuronul ascuns de la secvența de timp $t$, $y_t$ valoarea neuronului de ieșire la

timpul *t*, iar $x_t$ este valoarea neuronului de intrare la momentul *t,* $W_{xh}, W_{hh}, W_{hy}$ sunt matricile ponderilor dintre stratul de intrare și cel ascuns, dintre straturile ascunse, respectiv dintre stratul ascuns și cel de ieșire, $b_h$ și $b_y$ sunt valorile bias.

Adaptarea ponderilor se realizează prin implicarea algortimului de propagare înapoi a erorii adaptat pentru date temporale, denumit Backpropagation through time (propagarea înapoi a erorii în timp). Prin această metodă, rețeaua este desfășurată în secvențe de timp și se calculează gradienții pentru toți parametrii dintr-o întreagă secvență. La fel ca în cazul algoritmului backpropgation clasic, se determină diferența dintre valoarea prezisă (calculată) și cea dorită (reală), prin aplicarea funcțiilor din relațiile (7.20) - (7.23). Următoarea etapă este adaptarea ponderilor, care presupune realizarea următorii pași: (1) determinarea gradienților stratului de ieșire (8.3)-(8.4), (2) calcularea gradienților straturilor ascunse în raport cu ponderile (8.5)-(8.8), și (3) propagarea în timp (8.9).

$$\frac{\partial L}{\partial \hat{y}_t} = \frac{\partial L(y_t, \hat{y}_t)}{\partial \hat{y}_t} \tag{8.3}$$

Unde *L* este funcția de cost (care definește diferența dintre mărimea prezisă și cea reală), $y_t$ este ieșirea reală la momentul *t*, $\hat{y}_t$ este ieșirea prezisă la momentul *t*.

$$\frac{\partial L}{\partial V} = \sum_{t=1}^{T} \frac{\partial L}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial V} = \sum_{t=1}^{T} \frac{\partial L}{\partial \hat{y}_t} \cdot h_t^T \tag{8.4}$$

Unde relația determină gradientul față de ponderile *V*, *V* este matricea ponderilor dintre stratul ascuns și cel de ieșire, T este întregul interval.

$$\frac{\partial L}{\partial h_t} = \sum_{t'=t}^{T} \frac{\partial L}{\partial h_{t'}} \cdot \frac{\partial h_t'}{\partial h_t} \tag{8.5}$$

Unde relația determină gradientul funcției de cost în raport cu neuronii ascunși, $h_{t'}$ este valoarea din momentul de la secvența anterioară (astfel se evidențiază natura recurentă a rețelei).

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L}{\partial h_t} \cdot \frac{\partial h_t}{\partial W} = \sum_{t=1}^{T} \frac{\partial L}{\partial h_t} \cdot x_t^T \tag{8.6}$$

$$\frac{\partial L}{\partial U} = \sum_{t=1}^{T} \frac{\partial L}{\partial h_t} \cdot \frac{\partial h_t}{\partial U} = \sum_{t=1}^{T} \frac{\partial L}{\partial h_t} \cdot h_{t-1}^{T} \tag{8.7}$$

$$\frac{\partial L}{\partial b} = \sum_{t=1}^{T} \frac{\partial L}{\partial h_t} \tag{8.8}$$

Unde relația (8.6) determină gradientul față de ponderile *W*, *W* este matricea ponderilor dintre straturile ascunse, (8.7) determină gradientul funcției de cost față de *U*, *U* este matricea ponderilor dintre stratul de intrare și cel ascuns, (8.8) gradientul față de valoarea de bias.

Acești gradienți sunt propagați către ponderile dintre straturile ascunse, și în timp.

$$\frac{\partial L}{\partial h_{t-1}} = \sum_{t=1}^{T} \frac{\partial L}{\partial h_t} \cdot U \tag{8.9}$$

După acești pași este procesul propriuzis de adaptare a ponderilor cu relațiile (8.10) – (8.14).

$$W_{xh} = W_{xh} - \alpha \frac{\partial L}{\partial W_{xh}} \tag{8.10}$$

$$W_{hh} = W_{hh} - \alpha \frac{\partial L}{\partial W_{hh}} \tag{8.11}$$

$$W_{hy} = W_{hy} - \alpha \frac{\partial L}{\partial W_{hy}} \tag{8.12}$$

$$b_h = b_h - \alpha \frac{\partial L}{\partial b_h} \tag{8.13}$$

$$b_y = b_y - \alpha \frac{\partial L}{\partial b_y} \tag{8.14}$$

Unde $\alpha$ este rata de învățare.

Provocările în această abordare apar din cauză că în unele situații valorile ponderilor scad foarte mult (eveniment denumit dispariția ponderilor) sau cresc foarte mult (explozia ponderilor).

În categoria RNAR sunt incluse rețelele Elman, rețelele Jordan, rețelele Hopfield, rețelele de stare Ecou, memoria pe termen lung, unități recurente închise, și rețele neuronale recurente bidirecționale. Aceste rețele se diferențiază între ele după arhitectură și modul de funcționare astfel:

- Structura standard – dezavantajul principal constă în problemele de gradient a ponderilor.
- Rețeaua Elman – ciclul feedback se realizează între un strat ascuns și un strat context (se memorează secvența $h_{t-1}$).
- Rețeaua Jordan – ciclul feedback pornește de la stratul de ieșire, ci nu de la cel ascuns.
- Rețele bidirecționale – procesează secvențe de date atât înainte, cât și în direcția înapoi. În consecință, rezultatul final depinde atât de datele trecute, cât și de viitoare.
- Memoria pe termen lung (Long short-term memory) – a fost dezvoltată pentru a rezolva problema dispariției ponderilor apărută la rețelele recurente standard. Pentru aceasta, aceste rețele folosesc porți (gate) – poartă de intrare, poartă uită (forget gate) și poartă de ieșire.
- Unitatea recurentă închisă – este o variantă simplificată a memorie scurte pe termen lung, prin faptul că poarta de intrare și poarta uită sunt unificate într-o nouă poartă (update gate).
- Rețeaua Hopfield – este o rețea recurentă total interconectată, care poate lucra atât cu date discrete cât și continue.
- Rețeaua de stare Ecou – se caracterizează prin faptul că rezervorul de memorie este fix, și doar ponderile de la ieșire sunt adaptate.
- Modele de memorie diferențiale – extind RNAR prin adăugarea unor memorii externe pentru învățarea unor tipare mai complexe.
- Rețelele bazate pe atenție (Transformer RNAR).

Rețelele recurente folosite cu succes în energetică sunt rețelele standard, memoria scurtă pe termen lung, rețelele de stare Ecou, rețelele bidirecționale, rețelele de tip unitate recurentă închisă și rețelele transformator. Întrucât RNAR standard au fost deja prezentate, în continuare se vor descrie caracteristicile particulare ale celorlalte patru rețele neuronale recurente.

Memoria pe termen lung, MTL este o rețea recurentă particulară, care a fost dezvoltată pentru a elimina neajunsurile unei rețele recurente standard, adică dispariția sau explozia gradienților în procesul de antrenare folosind un interval lung de date. Pentru aceasta, MTL are celule de memorie și mecanisme de

deschidere, cu ajutor cărora rețeaua poate să memoreze legăturile temporale dintre secvențele de date și să își amintească dependențele pe termen lung. Elementul de bază a unei MTL este celula de memorie. Aceasta are capacitatea să își mențină starea pe o perioadă lungă de timp, ceea ce îi dă capabilitatea rețelei să își amintească evenimente trecute, și să facă predicții în funcție de date din trecutul apropiat, dar și îndepărtat. Structura unei celule de memorie a MTL este ilustrată în fig. 8.2. În plus, rețeaua are trei porți principale (intrare, ieșire și uită) prin care controlează circulația de date înspre interiorul și dinspre celula de memorie.

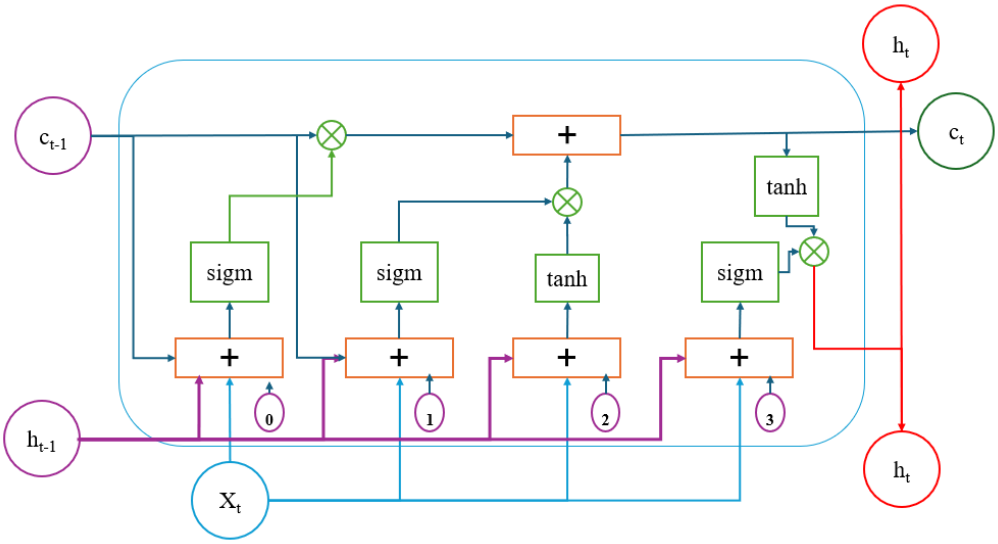



**Fig. 8.2.** Structura unei celule de memorie a MTL. $X_t$ intrarea, $c_{t-1}$ – memoria de la celula anterioară, $h_{t-1}$ ieșirea anterioară, $h_t$ – ieșirea, $c_t$ – memoria celulei, sigm – funcția sigmoidală, tanh – funcția tangentă hiperbolică, 0/1/2/3 – bias

Poarta de tip uită conectează memoria de la celula anterioară de celula curentă. Ea decide cât din memoria precedentă ajunge la celula curentă. Poarta uită este prima funcție de tip sigmoid (relația (8.15)) care se observă în fig. 8.2.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{8.15}$$

Unde $h_{t-1}$ este valoarea neuronului ascuns precedent, $x_t$ este intrarea la momentul $t$, $W_f$ este matricea ponderilor, iar $b_f$ este valoarea de bias.

Poarta de tip intrare determină cât din informația din celulă va fi păstrată. La fel ca în cazul precedent, este folosită funcția sigmoid (8.16) pentru a controla care informație se înmagazinează.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{8.16}$$

Iar pentru o nouă memorie care se poate adăuga este implicată funcția tangentă hiperbolică.

$$\tilde{c}_t = tanh(W_c[h_{t-1}, x_t] + b_c) \tag{8.17}$$

Poarta de tip ieșire influențează cât din memoria celulei va fi transmisă neuronului ascuns *h*. La fel ca în cazurile precedente, este folosită o funcție sigmoidală (8.18) pentru a regla memoria transmisă neuronului ascuns, iar valoarea acestuia este calculată implicând funcția tangentă hiperbolică (8.19).

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{8.18}$$

Iar valorea nodului ascuns este:

$$h_t = o_t \cdot \tanh(c_t) \tag{8.19}$$

Valoarea celulei *c* la momentul *t* se calculează în funcție de poarta uită, poarta de intrare, memoria celulei precedente și memoria de la momentul anterior.

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \tag{8.20}$$

În concluzie, avantajele rețelei MTL sunt următoarele:

- Dependențe pe termen lung - aceste rețele excelează la memorarea dependențelor pe termen lung, deoarece pot reține informații între secvențe lungi de date.
- Controlul folosind porți - porțile determină o memorie selectivă, ceea ce permite rețelei să se concentreze asupra datelor relevante și să le elimine pe cele irelevante.
- Prevenirea dispariției gradientului – MLT-urile sunt concepute pentru a atenua problema dispariției gradientului, făcându-le mai potrivite pentru antrenarea pe secvențe lungi de date.
- Eficiență pentru predicția secvenței - sunt utilizate pe scară largă în aplicațiile în care datele sunt secvențiale și prezintă dependențe temporale.

Rețeaua de stare Ecou este un tip special de rețea neuronală recurentă, care se caracterizează prin existența unui rezervor de unitărți de procesare. Această componentă cuprinde un număr mare de neuroni care sunt conectați aleator. O

caracteristică a metodei de antrenare, este faptul că la aceste rețele, ponderile neuronilor din rezervor rămân neschimbate, ci doar ponderile de la ieșire sunt modificate. Acest aspect face ca bagajul de calcule să scadă mult, făcând rețelele foarte folositoare în special în predicția datelor sub formă de serii de timp.

Precum se observă în fig. 8.3., rețeaua Ecou conține trei straturi de neuroni: stratul de intrare, stratul ascuns / rezervorul și stratul de ieșire.

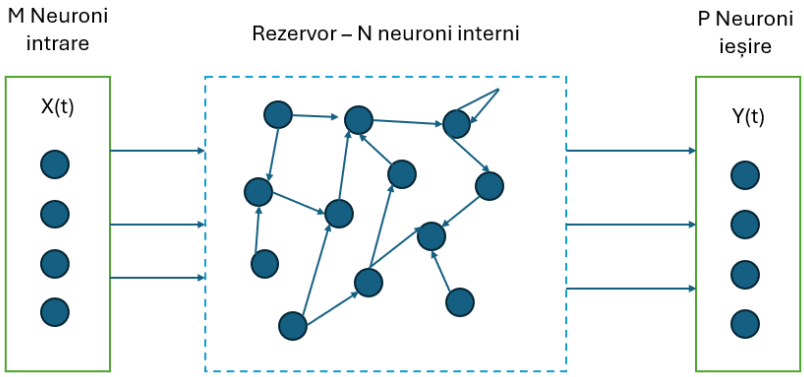



**Fig. 8.3.** Structura unei rețele recurente de stare Ecou

Stratul de intrare preia de la exterior, datele de intrare la fiecare moment de timp $t$ și le transmite neuronilor din rezervor. Acest nivel este o simplă poartă liniară care trimite datele mai departe. Astfel, neuronii de intrare realizează activarea de început a neuronilor din stratul ascuns, $R$.

$$r_t = f(W_{in}x_t + b_{in}) \tag{8.21}$$

Unde $r_t$ este starea / valoarea rezervorului la momentul $t$, $f$ este o funcție liniară, $W_{in}$ este matricea ponderilor dintre stratul de intrare și cel ascuns, $x_t$ sunt intrările, iar $b_{in}$ sunt valorile bias ale stratului de intrare.

Rezervorul, adică stratul ascuns, conține neuroni care au valorile ponderilor fixe la valorile de la început . Mai exact, ponderile dintre neuronii ascunși nu sunt ajustate, ci rămân la valoarea aleasă aleator la începutul procesului de antrenare a rețelei. În acest sens, ponderile ascunse aleatoare crează un ecou în timp a datelor de intrare, având capacitatea să recunoască și să învețe în mod dinamic tipare în datele de intrare. Neuronii rezervorului folosesc funcții de activare neliniare, precum funcția sigmoidală și tangentă hiperbolică. Starea rezervorului se modifică la fiecare secvență de timp astfel:

$$r_t = f(W_{res} r_{t-1} + W_{in} x_t + b_{res}) \tag{8.22}$$

Unde $r_{t-1}$ este starea / valoarea rezervorului la momentul *t-1*, *f* este o funcție neliniară de activare, $W_{res}$ este matricea ponderilor dintre neuronii ascunși, care are o valoare fixă, $x_t$ sunt intrările, iar $b_{res}$ sunt valorile bias ale stratului ascuns.

Matricea ponderilor neuronilor ascunși influențează modul de propagare a ecoului datelor de intrare.

Stratul de ieșire furnizează ieșirea rețelei în funcție de starea rezervorului.

$$y_t = W_{out} \cdot r_t \tag{8.23}$$

Unde $r_t$ este starea / valoarea rezervorului la momentul prezent *t*, $W_{out}$ este matricea ponderilor de ieșire dintre stratul ascuns și cel de ieșire.

Ponderile de ieșire sunt singurele care se ajustează în procesul de antrenare a rețelei. Tehnica cea mai folosită se bazează pe regresie (regresia crestei sau regresia celor mai mici pătrate), care dă rezultate mai rapide decât metoda backpropagation care se folosește la RNAR standard. Ajustarea ponderilor de ieșire prin implicarea regresiei se face prin relația (8.24).

$$W_{out} = (R^T R + \alpha I)^{-1} R^T Y \tag{8.24}$$

Unde *R* este matricea ponderilor dintre neuroni ascunși memorată în timp, *I* este matricea de identitate, *Y* este matricea valorilor dorite (reale), iar $\alpha > 0$ este parametru de regularizare.

În concluzie, rețeaua de stare Ecou este o variantă eficientă a rețelelor neuronale recurente, care prin rezervorul fix aleatoriu permite să se modeleze dinamica temporală complexă fără neajunsul care presupune un bagaj de calcul în situația antrenării tuturor ponderilor. Ea excelează în sarcini care implică predicția datelor temporale, sisteme dinamice și date secvențiale, făcându-le potrivite pentru aplicații precum sistemele de prognoză și control.

Rețelele recurente bidirecționale sunt o extensie a RNAR standard. Astfel, o rețea bidirecțională conține două rețele recurente – prima realizează procesarea datelor de intrare în direcția strat inferior – strat superior (1->T), iar a doua procesează datele în sens invers, de la ieșire spre intrare (T->1); iar la fiecare moment, ambele valori ascunse sunt folosite pentru a determina ieșirea. Baza matematică din spatele funcționării acestui tip de rețea este descrisă prin relațiile (8.25) – (8.27).

$$\overrightarrow{h_t} = f(W_{\overrightarrow{xh}}x_t + W_{\overrightarrow{hh}}\vec{h}_{t-1} + b^{\rightarrow}) \quad (8.25)$$

$$\overleftarrow{h_t} = f(W_{\overleftarrow{xh}}x_t + W_{\overleftarrow{hh}}\overleftarrow{h_{t+1}} + b^{\leftarrow}) \quad (8.26)$$

$$y_t = g(W_{hy}[\overrightarrow{h_t}; \overleftarrow{h_t}] + b_y) \quad (8.27)$$

Unde *f* de obicei este funcția sigm sau tanh, *g* este funcția softmax, *x* este șirul datelor de intrare, *y* este ieșirea, $\overrightarrow{h_t}$.este valoarea nodului ascuns când informația se propagă înainte, $\overleftarrow{h_t}$.este valoarea nodului ascuns când informația se propagă înapoi, *W* sunt matricile ponderilor dintre intrare și stratul ascuns, dintre straturile ascunse, respectiv dintre stratul ascuns și cel de ieșire.

Aceste rețele dau rezultate foarte bune în clasificare, dar nu pot fi folosite în timp real, deoarece este nevoie de la început de toată secvența de date.

Unitatea recurentă închisă, URI este o variantă simplificată a memoriei pe termen lung, dar care păstrează performanțele acesteia, iar în unele situații dă rezultate mai bune. Astfel, o URI combină porțile de uitare și de intrare într-o singură poartă de actualizare și îmbină starea celulei de memorie și starea ascunsă într-o singură unitate. Structura unei celule de memorie a unei URI este ilustrată în fig. 8.4. Se poate observa în structura celulei de memorie poarta de actualizare și poarta de resetare, care folosesc pentru activare funcția sigmoid, iar pentru determinarea valorii noi a memoriei funcția tangentă hiperbolică.

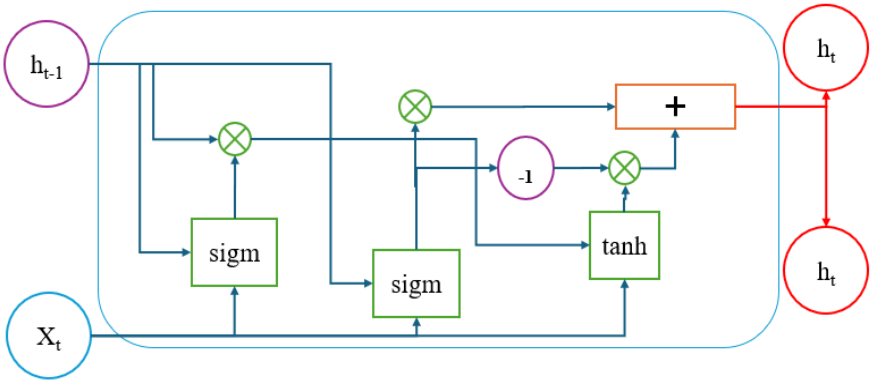



**Fig. 8.4.** Structura unei celule de memorie a URI.

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \quad (8.28)$$

Unde $h_{t-1}$ este valoarea neuronului ascuns precedent, $x_t$ este intrarea la momentul *t*, $W_z$ și *U* este matricea ponderilor, iar $b_z$ este valoarea de bias.
Determină cât din informația din trecut să se păstreze.

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{8.29}$$

Determină cât din informația din trecut să se uite.

$$\tilde{h}_t = tanh(W_h x_t + U_h(r_t * h_{t-1}) + b_h) \tag{8.30}$$

* este operația de înmulțire în funcție de înmulțire.
Valoarea nouă a memoriei.

$$h_t = (1 - z_t) * h_{t-1} * \tilde{h}_t \tag{8.31}$$

Valoarea nodului din stratul ascuns.

Avantajele URI față de MTL sunt următoarele:

- Antrenament mai rapid întrucât sunt mai puține porți și mai puțini parametri.
- Structură simplă, mai ușor de implementat.
- Bun pentru seturi de date mici în care MTL s-ar putea supraadapta.
- Eficient în determinarea dependențelor pe secvențe moderate până la lungi de date temporale.

### 8.3.Rețele competitive

Rețelele neuronale artificiale competitive, RNAC denumite și rețele cu învățare competitivă, fac parte din categoria RNA cu învățare nesupravegheată, în timpul căreia neuronii concurează între ei pentru a se activa, și nu colaborează precum în cazul învățării supravegheate. Caracteristic învățării nesupravegheate este existența unui baze de date de antrenament neordonate, și faptul că neuronii învață prin competiție, astfel la un moment dat, doar un singur neuron (sau un grup mic de neuroni) se va activa, iar ponderile corespunzătoare lui vor fi ajustate. Aceasta este strategia ”câștigătorul ia tot”.

În literatura de specialitate există multe exemple de reţele cu învăţare competitivă, însă prima variantă a fost introdusă de Rumelhart şi Zipper în 1986. În continuare, se vor prezenta caracteristicile acestei rețele competitive, care stă la baza tuturor variantelor de RNAC care i-au urmat.

Rețelele cu învăţare competitivă folosesc un nivel de unităţi de procesare (neuronii din stratul de ieșire) în care unităţile concurează una cu cealaltă, rezultatul fiind o reţea care poate fi folosită în aplicaţii de clasificare a structurilor, fig. 8.5.

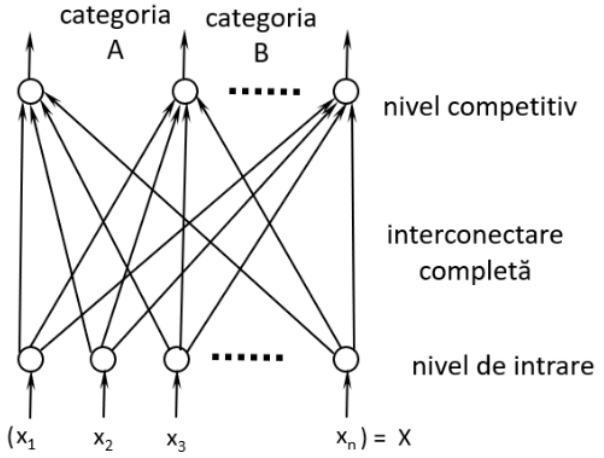



**Fig. 8.5.** Arhitectura unei rețele neuronale competitive

Competiţia poate fi desăvârşită cu ajutorul unui algoritm care desemnează unitatea învingătoare sau ca o alternativă, prin inhibare între unităţile din nivelul competitiv. În cazul inhibării, nivelul competitiv progresează într-o stare în care numai o unitate învingătoare este activă.

În procesul de antrenare, rețelei i se prezintă un set de structuri de antrenare, dar nu i se oferă un răspuns „ţintă” pentru fiecare structură de intrare. Reţeaua organizează structurile de antrenare într-un set de clase proprii. În nivelul competitiv unităţile concurează pentru ocazia de a răspunde structurii de intrare, iar câştigătorul reprezintă categoria de clasificare pentru structura respectivă de intrare.

Din arhitectura rețelei, reiese că rețeaua cuprinde două straturi de neuroni, stratul de intrare și stratul de ieșire sau competitiv. Neuronii acestor straturi sunt complet interconectați. Neuronii de intrare preiua informația de la exterior și o transmit neuronilor competitivi prin intermediul ponderilor care îi leagă. Vectorul de intrare este liniar, componentele sale luând valori în mulţimea {0,1}.

Neuronii competitivi vor realiza clasificarea, iar cel care corespunde datei dorite va fi activat și va oferi ieșirea.

Ponderile au valori limitate în intervalul [0,1]. Acestea însă trebuie să îndeplinească condiția ca suma ponderilor asociate fiecărei unităţi din nivelul competitiv este egală cu unu.

Iniţial, în prima etapă a procesului de antrenare, pentru ponderi se aleg valori aleatoare care îndeplinesc condiţia amintită mai sus.

$$\sum_{j=1}^{n} w_{ij} = 1 \tag{8.31}$$

Următoarea etapă în procesul de antrenare constă în realizarea următorilor doi pași:

1. Se calculează suma ponderată pentru fiecare unitate de nivel competitiv

$$S_i = \sum_{j=1}^{n} w_{ij} x_j \tag{8.32}$$

2. Are loc competiţia între unităţile de ieşire. Prin abordarea „câştigătorul – ia - tot", unitatea cu cea mai mare sumă ponderată este desemnată ca învingătoare. Acestei unităţi i se atribuie o nouă valoare de activare egală cu „1"şi toate celelalte unităţi se setează la valoarea „0". Dacă apare o situaţie de egalitate $S_j = S_i$ atunci prin convenţie, unitatea din stânga va fi selectată.

$$u_j = \begin{cases} 1, S_j > S_i \\ 0, S_j \leq S_i \end{cases} \tag{8.33}$$

Ponderile sunt ajustate după ce „câştigătorul" este desemnat prin utilizarea relației (8.34).

$$\Delta w_{ji} = g \cdot \left( \frac{x_i}{m} - w_{ji} \right) \tag{8.34}$$

Unde $g$ este un parametru de învăţare (ales de utilizator), $m$ este numărul de unităţi din nivelul de intrare care au niveluri de activare 1, $x_i$ este valoarea unităţii $i$.

Parametrul $g$ reflectă "cantitatea" de ajustare a ponderilor la fiecare iteraţie; tipic $g = 0{,}01 \ldots 0{,}03$. În relația (8.34) primul termen din relaţie determină ca ponderea să fie incrementată când unitatea de intrare corespunzătoare este activă $x_i = 1$ şi să fie decrementată când unitatea este inactivă $x_i = 0$. Ponderile devin mai apropiate de valoarea unităţii respective de intrare după reiniţializare. Ajustarea din relație face ca suma ponderată pentru fiecare structură de intrare să fie uşor mai mare dacă aceeaşi structură de intrare este prezentată imediat, din nou reţelei. În plus, tipare din datele de intrare similare cu tiparul datelor de

intrare de la momentul analizat vor produce de asemenea o sumă ponderată mai mare pentru unitatea învingătoare.

În categoria RNAC intră rețelele competitive standard, descrise anterior, hărțile auto-organizate (SOMs – Self-Organizins Maps), învățarea cuantizării vectoriale (LVQ - Learning Vector Quantization), rețelele ”winner takes all” și teoria rezonanței adaptive. Dintre acestea, SOMs și LVQs sunt cele mai folosite pentru dezvoltarea de aplicații dedicate rezolvării problemelor asociate sistemelor electroenergetice.

Hărțile auto-organizate, denumite și rețele Kohonen au caracteristicile rețelelor competitive prezentate anterior, precum învățarea competitivă și structura bi-nivel (strat de intrare și strat competitiv) și vectorul de ponderi asociat fiecărui neuron. Particular rețelelor Kohonen este faptul că stratul de ieșire poate fi uni (1D, vector) sau bi-dimensional (2D, matrice), așa cum este arătat în fig. 8.6. În plus, nu doar neuronul câștigător beneficiază de ajustarea ponderilor și activare, dar și neuronii învecinați lui, aceasta deoarece neuronii concurează pentru a câștiga, dar neuronii învecinați colaborează. În acest fel, fiecare neuron din stratul competitiv ajunge să reprezinte un vector prototip corespunzător unor caracteristici a datelor de intrare.

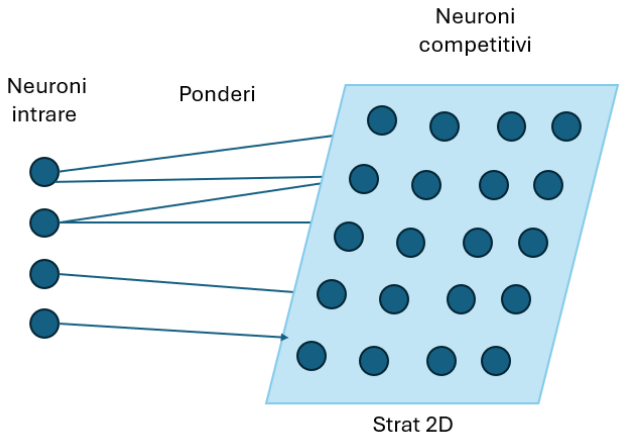



**Fig. 8.6.** Arhitectura unei rețele Kohonen (2D)

Baza matematică a acestor rețele cuprinde relații deja prezentate, însă modificate pentru a corespunde rețelelor Kohonen, plus relații matematice particulare. Pentru a înțelege, este necesară descrierea pașilor prin care trec datele în interiorul acestor rețele. Astfel, primul pas este inițializarea ponderilor cu valori aleatoare, urmată de determinarea valorii neuronului *i*, care este cel mai aproape de vectorul de intrare.

$$i^* = \arg min \|\vec{x} - \vec{w}_i\| \tag{8.35}$$

În plus, este definită o funcție de vecinătate $h_{i,i^*}(t)$, care scade cu distanța față de neuronul învingător și în timp. O funcție de vecinătate des întâlnită este funcția gaussiană de vecinătate.

$$h_{i,i^*}(t) = exp\left(-\frac{\|r_i - r_{i^*}\|^2}{2\sigma(t)^2}\right) \tag{8.36}$$

unde $r_i$ este poziția neuronului în stratul 2D, iar $\sigma(t)$ este lățimea vecinătății care se micșorează în timp.

La următorul pas, se actualizează valorile ponderilor neuronului învingător *i* și a vecinilor. În acest fel, vectorul corespunzător învingătorului *i* se va apropia mai mult de vectorul de intrare.

$$\vec{w}_i(t+1) = \vec{w}_i(t) + \alpha(t) \cdot h_{i,i^*}(t) \cdot (\vec{x} - \vec{w}_i(t)) \tag{8.37}$$

Unde $\alpha(t)$ este rata de învățare, care se reduce în timp pentru a realiza convergența spre o valoare minimă.

Acești pași se repetă pentru toate datele de intrare și pentru mai multe epoci, până se atinge condiția de oprire. În timpul procesului de antrenare, pentru a crea condiția de convergență, valorile ratei de învățare și a lățimii vecinătății se reduc.

Avantajele acestui tip de rețea competitivă provin de la învățarea nesupravegheată, topologia spațială a neuronilor, și numărul redus de neuroni și conexiuni, ceea ce fac aceste rețele adecvate pentru gruparea datelor, extragerea caracteristicilor și robuste la zgomote. Aceste puncte forte sunt însă umbrite de dezavantajele lor – convergență încetinită, granițe neclare între clase, nu dau rezultate bune pentru datele temporale și greu de interpretat dacă sunt date multe.

Învățarea cuantizării vectoriale este un algoritm asociat cu rețelele competitive de tip SOM, producând rețelele competitive LVQ.

Algoritmul de învățare a cuantizării vectoriale folosește abordarea supravegheată, care implică aceeași strategie competitivă ca în cazul rețelelor SOM, dar în plus se adaugă etichete claselor pentru a ghida procesul de antrenare. Față de rețelele SOM, aceste rețele sunt utilizate în special pentru clasificare, dar comparativ cu primele, acestea sunt mai simple, interpretabile și eficiente.

Structura unei rețele LVQ cuprinde stratul de intrare și stratul de ieșire, care cuprinde vectori ponderați asociați unei clase etichetate cunoscute. Prima etapă în antrenarea rețelei este inițializarea fiecărui vector ponderat cu o etichetă a unei

clase. În următoarea etapă se determină apropierea vectorului de intrare de vectorul de ieșire etichetat cu ajutorul relației (8.35).

Dacă se presupune că $c_x$ este clasa vectorului de intrare x, iar $c_{i^*}$ este clasa vectorului câștigător, atunci în situația în care comparația arată o asemănare, se va muta vectorul înspre caracteristicile datelor de la intrare

$$\vec{w}_{i^*} = \vec{w}_{i^*} + \alpha(t) \cdot (\vec{x} - \vec{w}_{i^*}) \tag{8.38}$$

În caz contrar, se mută vectorul mai departe de caracteristicile datelor de la intrare.

$$\vec{w}_{i^*} = \vec{w}_{i^*} - \alpha(t) \cdot (\vec{x} - \vec{w}_{i^*}) \tag{8.39}$$

Acest tip de rețea prezintă aceleași puncte forte și slabe ca rețelele SOM, dar în plus față de acestea sunt mai simple, mai rapid de antrenat și pot fi folosite pentru clasificarea cu multe categorii.

### 8.4.Rețele antrenate prin întărire

Rețelele antrenate prin întărire (reinforcement learning cu RNA) diferă față de rețelele prezentate anterior prin faptul că includ în funcționarea lor un algoritm de învățare prin întărire.

Învățarea prin întărire / consolidare este un tip de învățare automată, în care un agent învață să ia decizii prin interacțiunea cu un mediu. Agentul selectează acțiuni în funcție de starea mediului la un moment dat, și va primi un răspuns, care poate fi o recompensă numerică și o nouă stare. Obiectivul agentului este de a învăța o strategie /politică prin care se maximizează recompensa cumulativă în timp, însă echilibrând explorarea și exploatarea. Explorarea se referă la încercarea de noi acțiuni, iar exploatarea este alegerea celei mai cunoscute acțiuni.

Componentele unui sistem care se bazează pe învățarea prin întărire cuprinde: agentul, mediul cu care intercaționează agentul, starea mediului la un moment dat, acțiunea aleasă și realizată de agent, recompensa primită de agent și strategia / politica folosită de agent, care este o funcție reprezentată de RNA. Astfel, rețeaua neuronală artificială are rolul de aproximator al funcției, și pe baza algoritmului de învățare prin întărire abordat, aceasta va aproxima valoarea Q a funcției, politica și valoarea funcție, folosind ponderile și valorile bias.

Agentul reprezintă un algoritm care ia toate deciziile pe baza interacțiunii cu mediul, astfel, acesta observă stările mediului, alege acțiunile și învață condiționat de recompense. El în cazul abordărilor care folosesc metodologia de

alegere bazată pe actor-critic, cupinde o componentă denumită actor, care se ocupă exclusiv cu alegerea acțiunilor în funcție de politica aleasă.

Mediul reprezintă tot ce se află în afara agentului și interacționează cu acesta. Astfel, în aplicațiile dedicate sistemelor electroenergetice, acesta cuprinde toate sistemele fizile și operaționale pe care agentul încearcă să le controleze sau optimiza. Mediul este caracterizat prin spațiul stărilor, spațiul acțiunilor, funcția recompensei și modificarea dinamicii după fiecare acțiune.

Spațiul stărilor se referă tot ceea ce agentul observă la un moment dat, și depinde de aplicație. De exemplu, o aplicație dedicată controlului microrețelor are în spațiul stărilor informații despre producția de energie, cererea de energie și starea de încărcare a sistemului de stocare a energiei.

Spațiul acțiunilor reprezintă tot ceea ce poate agentul face. Astfel, o aplicație care realizează controlul frecvenței, în spațiul acțiunilor are acțiunile: modifică plotul transformatorului, modifică treapta bateriei de condensatoare. Acțiuniile pot fi discrete sau continue.

Funcția recompensei (notată $R$(s; a), a – acțiune, s - stare) arată cum agentul va fi evaluat. Această funcție are diferite forme, care depind de obiectivele aplicației. Aceste obiective pot fi operaționale (de exemplu, menținerea frecvenței), economice (minimizarea costurilor) sau constrângeri de siguranță și fiabilitate.

Modificarea dinamicii sistemului arată modul în care acesta evoluează după fiecare acțiune a agentului.

Matematic, obiectivul algoritmului care este maximizarea recompensei cumulate, care se poate exprima astfel:

$$J(\theta) = E_{\pi_0}[\sum_{t=0}^{\infty} \gamma^t r_t] \tag{8.40}$$

unde $\theta$ reprezintă parametrii rețelei, $\gamma \in [0,1]$ este factorul de reducere, $r_t$ este recompensa la momentul $t$, iar $\pi_0$ este politica parametrizată de ponderile rețelei.

Abordarea prin care se aproximează valoarea $Q$ presupune utilizarea relațiilor matematice (8.41) și (8.42).

$$Q(s,a) = E[r_t + \gamma\, maxQ(s',a')] \tag{8.41}$$

unde $Q$ reprezintă funcția acțiune, iar rețeaua este antrenată să aproximeze funcția de cost / pierdere

$$L(\theta) = (y - Q(s,a;\theta))^2 \tag{8.42}$$

unde $y = r - \gamma \, max_{a`} Q(s', a'; \theta^-)$, iar $\theta^-$ este rețeaua țintă (valorile ponderilor și a bias spre care converge algoritmul).

Abordarea prin aproximarea politicii folosite, presupune optimizarea directă a metodei de alegere a acțiunilor prin implicarea metodologiei creșterea gradientului:

$$\nabla_\theta J(\theta) = E_{\pi_0}[\nabla_\theta log\pi(a|s;\theta) \cdot R] \tag{8.43}$$

unde rețeaua reprezintă politica de alegere a acțiunilor.

O altă abordare este să se utilizeze metode care se bazează pe actor. Prin această abordare se pot urma două direcții: direcția actorului (funcția $\pi(a|s;\theta)$) sau direcția criticului (valoarea funcție $V(s;\varphi)$.

Actualizările realizate asupra actorului (algoritmul de selecție a acțiunilor) sunt ghidate de răspunsul criticului (estimarea avantajului):

$$\nabla_\theta J(\theta) = \nabla_\theta log\pi(a|s;\theta) \cdot A(s,a) \tag{8.44}$$

Rețelele neuronale care sunt combinate cu învățarea prin întărire și folosite pentru aplicații în inginerie energetică sunt rețelele de tip multistrat perceptron (rețele feedforward), rețelele recurente și rețelele convoluționale. În aceste situații, structura de bază a algoritmului rămâne neschimbată, dar arhitectura rețelelor modelează reprezentarea datelor de intrare, modul în care sunt utilizate datele anterioare și afectează complexitatea și puterea de rezolvare a politicii și estimarea valorii.

Caracteristicile rețelelor menționate coroborate cu învățarea prin întărire sunt următoarele:

- Rețele multistrat perceptron – aceeași arhitectură și oferă rezultate bune în cazul mediilor unde fiecare stare este inedependentă. Algoritmul de învățare principal este tot backpropagation, unde stările sunt tratate ca un vector al trăsăturilor de valori fixe. În continuare, rețeaua nu poate lucra eficient cu date temporale.
- Rețele recurente – se păstrează memoria internă și arhitectura rețelei, dar în plus aceasta poate lucra cu stări secvențiale. Datele de intrare se schimbă, întrucât este nevoie atât de memoria internă cât și starea curentă, dar algoritmul de învățare principal se menține, adică backpropagation în timp. Noile sisteme de învățare prin întărire sau capabilitatea agentului să ”înțeleagă” contextul într-o manieră temporală. Aceste rețele dau rezultate foarte bune acolo unde datele trecute au o mare importanță.

- Rețele convoluționale – caracteristic acestei combinații este faptul că sunt folosite atunci când starea este bazată pe imagini (pentru a extrage caracteristici spațiale), astfel datele de intrare sunt sub forma unor tensori 2D sau 3D. Diferențele care apar în algoritmul de învățare prin întărire sunt straturile de convoluție și grupare care sunt adăugate înaintea straturilor complet conectate; hărțile de caracteristici sunt extrase și transmise mai departe conducătorilor de politici / valori.
- Rețele memorie pe termen lung – aceste rețele antrenate prin algoritmul studiat sunt eficiente pentru medii dinamice și complexe. Dezavantajul este că existența celulelor de memorie și a porților va îngreuna procesul de antrenare printr-un cost computațional mai ridicat, dar adesea duce la politici mai robuste în sarcini variate în timp.

### 8.5.Rețele spiking

Rețelele neuronale spiking, RNS fac parte din RNA de generația a treia (perceptronul – generația 1, multistrat perceptronul, rețelele convoluționale și recurente – generația 2), care imită fidel dinamica temporală a rețelelor neuronale și a neuronilor biologici. Astfel, comparativ cu rețelele prezentate anterior a căror neuroni folosesc activarea continuă prin utilizarea unor funcții bine definite care mereu vor furniza valori, neuronii spiking transmit informații mai departe doar la momentul în care activarea a ajuns la un anumit prag. Această activare se realizează sub forma unui spike (vârf discret / creste) la anumite momente în timp. Analogia cu neuronii biologici este următoarea: rețeaua așteaptă până când potențialul de membrană al unui neuron atinge un prag și apoi emite un vârf, la fel ca neuronii din creier.

Arhitectura unei RNS este la fel ca a unei rețea feedforward, sau a unei rețele recurente, adică au un strat de intrare, straturi ascunse și un strat de ieșire. Legătura dintre neuroni se face prin intermediul conexiunilor sinaptice, adică a ponderilor care se ajustează în timpul procesului de antrenare. La fel ca în cazul RNA de generația a 2-a, neuronii trec printr-o etapă de activare. Diferența majoră dintre rețelele precedente și RNS este modul în care sunt activați neuronii și antrenarea, care se face prin implicarea algoritmului de gradiente surogat.

Elementele definitorii ale RNS sunt spike-urile (vârfurile – semnale discrete trimise între neuroni), potențialul de membrană (fiecare neuron are o stare internă care acumulează vârfuri de intrare), pragul de activare (când potențialul membranei depășește această valoare, neuronul este activat), timpul de întârziere

(durata de timp cât îi trebuie unui vârf să ajungă la următorul neuron), și perioada refractară (timpul după activare / declanșare în care neuronul nu poate să crească din nou).

Se observă din paragraful anterior că toate caracteristicile unei rețele spiking se referă la trăsăturile neuronilor din care sunt alcătuite. Trei modele neuronale care alcătuiesc RNS sunt neuronul LIF (Leaky Integrate-and-Fire), neuronul Izhikevich și neuronul Hodgkin-Huxley.

Neuronul LIF este cel mai des folosit, dar are un punct slab, și anume potențialul membranei unui neuron se degradează în timp, dacă nu este întărit de vârfuri.

Neuronul LIF este modelat prin utilizarea unei ecuații diferențiale cu ajutorul căreia se memorează dinamica potențialului membranei în timp. Dinamica potențialului membranei se determină prin implicarea relației (diferențială de ordinul 1)

$$\tau_m \frac{dV(t)}{dt} = -[V(t) - V_{res}] + R \cdot I(t) \tag{8.45}$$

unde $V(t)$ este potențialul membranei la momentul *t*, $\tau_m$ este constanta de timp a membranei, $V_{res}$ este potențialul aflat în așteptare, $R$ este rezistența membranei, iar $I(t)$ este curentul de intrare.

Potențilul membranei, $V(t)$ a unui neuron, când ajunge la o anumită valoare de prag, $V_{\text{th}}$, neuronul va declanșa un semnal de vârf, apoi potențialul este resetat la valoarea de resetare, $V_{reset}$, iar apoi neuronul poate intra într-o perioadă refractară în care neuronul nu va putea declanșa nici un semnal de vârf.

În situațiile în care neuronul lucrează cu semnale discrete, atunci relația anterioară se va modifica astfel

$$V[t+1] = V[t] + \frac{\Delta t}{\tau_m}(-[V[t] - V_{res}] + R \cdot I[t]) \tag{8.46}$$

Dacă se respectă condiția ca $V[t+1] \geq V_{th}$, atunci neuronul va emite un semnal vârf și se va reseta la o valoare predefinită $V[t+1] \leftarrow V_{reset}$.

Funcționarea neuronului se poate explica astfel: neuronul acumulează în timp curent de la intrare, care în timp va crește potențialul membranei. Dacă nu primește curent, potențialul neuronului se degradează în timp și ajunge la potențialul de așteptare (se simulează comportamentul neuronului biologic, care pierde ioni dacă nu este activat). Dacă neuronul acumulează suficient potențial pentru a atinge valoarea de prag, va declanșa un semnal de vârf și apoi se resetează și intră în starea refractară, neputând fi accesat temporat.

Când un neuron primește informații de la mai mulți neuroni prin conesiunile sinaptice, datele de intrare sub formă de semnal de vârf, atunci intrarea neuronului, $I(t)$ se determină folosind relația matematică, care îi dă capacitatea neuronului LIF să răspundă la intrările temporale :

$$I(t) = \sum_j w_i \cdot \delta\,(t - t_j) \tag{8.47}$$

unde $w_i$ sunt ponderile, $t_j$ sunt timpi semnalelor sinaptice, iar $\delta(t - t_j)$ este funcția delta Dirac, care reprezintă semnalurile vârf.

Neuronul Izhikevich reușește să echilibreze foarte bine realismul biologic și eficiența computațională. Acest neuron este caracterizat printr-un sistem cu două variabile.

$$\begin{cases} \frac{dv}{dt} = 0{,}04v^2 + 5v + 140 - u + I \\ \frac{du}{dt} = a(bv - u) \end{cases} \tag{8.48}$$

Având condiția de resetare și declanșare a neuronului.

$$Dacă\ v \geq 30\ mV, Atunci\ \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{8.49}$$

unde *I* este curentul de intrare, *v* este potențialul membranei, *u* este variabila care caracterizează recuperarea membranei, *a* – scala de timp a lui *u*, *b* – sensibilitatea lui *u* față de *v*, *c* – resetează valoarea lui *v* după declanșare, *d* – resetează incrementul lui *u*.

Funcționarea neuronului Izhikevich se desfășoară pe o durată de timp în care sunt realizați patru pași:

1. Variabila *v* simulează potențialul membranei și integrează curentul de intrare, prin (8.48).
2. Variabila *u* acționează pentru a realiza recuperarea sau adaptarea, precum feedback-ul inhibitor. Această variabila de recuperare ajută la modelarea momentului și tipului de vârfuri.
3. Termenul neliniar $v^2$ +5*v*+140 permite modele complexe de declanșare (de exemplu, declanșare explosivă, declanșare rapidă).
4. Când *v* atinge 30 mV (spike hardcoded), neuronul se declanșează și se resetează.

Avantajul utilizării acestui model neuronal constă în faptul că poate simula comportamentul majorității tipurilor de neuroni declanșați prin semnale de vârf, aceasta prin setarea corespunzătoare a parametrilor *a*, *b*, *c*, *d*.

Neuronul Hodgkin-Huxley este foarte precis din punct de vedere biologic, dar costisitor din punct de vedere informatic. Acest neuron este caracterizat prin relația matematică (8.50), la care se adaugă trei variabile de intrare (de trecere / gating) (8.51).

$$C_m \frac{dV}{dt} = I - \bar{g}_{Na} m^3 h (V - E_{Na}) - \bar{g}_K n^4 (V - E_K) - \bar{g}_L (V - E_L) \quad (8.50)$$

$$\begin{cases} \frac{dm}{dt} = \alpha_m(V)(1-h) - \beta_m(V)h \\ \frac{dh}{dt} = \alpha_h(V)(1-h) - \beta_h(V)h \\ \frac{dn}{dt} = \alpha_n(V)(1-h) - \beta_{hn}(V)h \end{cases} \quad (8.51)$$

Unde $V$ este potențialul membranei, $C_m$ este capacitatea membranei, $I$ - curentul de intrare, $g_x$ este conductanța maximă a canalul ionului, $x$ = K, Na, Leak - scurgere, $E_x$ – potențialul de inversare pentru neuronul $x$, $m$, $n$, h - variabile de intrare pentru canalele de sodiu (activare/inactivare) și potasiu, $\alpha, \beta$ – funcții de rată dependente de tensiune.

Acest model neuronal este cel mai apropiat de neuronul biologic, astfel el simulează curenții ionici care curg prin canalele de sodiu, potasiu și scurgerile în membrana unui neuron biologic. În plus, porțile de intrare dependente de tensiune se deschid / se închid cu probabilități specifice bazate pe tensiune. În consecință, combinația din dinamica canalului face ca potențialul de acțiune să crească și să scadă — foarte aproape de ceea ce se întâmplă în biologie. Rezultă forme de undă de vârf (spike) realiste, cu sincronizare și formă precise.

O sinteză a caracteristicilor de bază a celor trei neuroni artificiali prezentați este prezentată în tabelul 8.2.

**Tabelul 8.2.** Comparație între neuronii spike

| **Caracteristică** | **Neuron LIF** | **Neuron Izhikevich** | **Neuron Hodgkin-Huxley** |
|---|---|---|---|
| Complexitate | Scăzută | Medie | Mare |
| Realism biologic | Scăzută | Mare | Foarte mare |
| Viteza de simulare | Foarte rapidă | Rapidă | Lentă |

| | | | |
|---|---|---|---|
| Tip de declanșare | Limitată | Bogată | Realistă, dar greu de setat |
| Utilizare | Rețele spiking, implementare fizică | RNA similar creierul uman | Neurologie |

Un alt aspect important la rețelele spiking este algoritmul de învățare. Întrucât acestea pot fi antrenate atât prin implicarea învățării supravegheate și nesupravegheate, în continuare se vor prezenta cei doi algoritmi, adaptați pentru aceste rețele.

Antrenarea prin învățare supravegheată se realizează prin folosirea algoritmului micșorarea gradientului surogat. Deoarece funcția de tip vârf nu este diferențiabilă, metodele de tip gradientului surogat aproximează derivata funcției în timpul propagării înapoi a erorii. Baza matematică a algoritmului este definită în continuare.

Se presupune funcția de vârf ca fiind funcția de pas Heaviside

$$S(t) = H(V(t) - V_{th}. \tag{8.52}$$

Această funcție nu are gradient, dar acesta se va înlocui cu

$$\frac{\partial S}{\partial V} \approx \varphi(V). \tag{8.53}$$

Unde acest gradient surogat poate căpăta forma din (8.54) (derivata funcției sigmoid) sau (8.55) (estimatorul direct).

$$\varphi(V) = \frac{1}{1+e^{-\beta(V-V_{th})}}\left(1 - \frac{1}{1+e^{-\beta(V-V_{th})}}\right). \tag{8.54}$$

$$\varphi(V) = \begin{cases} 1, dacă\ |V - V_{th}| < \epsilon \\ 0, în\ caz\ contrar \end{cases}. \tag{8.55}$$

Următorul pas este utilizarea algoritmului de backpropagation în timp pentru a ajusta ponderile dintre neuroni. În concluzie, în procesul de învățare, informația se propagă de la intrare spre ieșire precum într-o RNS, iar propagarea înpoi a erorii dinspre ieșire spre intrare precum într-o RNAF, unde se presupune că funcția vârf este diferențiabilă prin utilizarea unei aproximații.

Antrenarea prin învățare nesupravegheată este poate realiza prin algoritmul denumit plasticitatea dependentă de spike-timing (momentul vârfului). Prin acest

algoritm, ponderile sunt modificate pe baza momentului relativ al vârfurilor între neuronii pre- și post-sinaptici.

$$\Delta w = \begin{cases} A_+ \cdot e^{-\frac{\Delta t}{\tau_+}}, dacă\ \Delta t > 0,\ \ pre\ înainte\ de\ post \\ -A_- \cdot e^{-\frac{\Delta t}{\tau_-}}, dacă\ \Delta t < 0,\ \ post\ înainte\ de\ pre \end{cases} \quad .(8.56)$$

Unde $\Delta t = t_{post} - t_{pre}$, $A_+, A_-$ sunt ratele de învățare, $\Delta w$ – modificarea cu care sunt ajustate ponderile, $\tau_+, \tau_-$ - constante de timp pentru potențare și depresie.

Algoritmul funcționează astfel: dacă un neuron presinaptic crește chiar înaintea unui neuron postsinaptic, sinapsa este întărită (învățare Hebbian), iar dacă vârful presinaptic este după vârful postsinaptic, sinapsa este slăbită. În concluzie, acest algoritm permite RNS să învețe tipare temporale într-un mod nesupravegheat.

Avantajele RNS legate de prelucrarea temporală a informațiilor, eficiența energetică, plauzibilitatea biologică, putere de calcul mare și compatibilitate cu senzori bazați pe evenimente. Într-adevăr, RNS pot captura sincronizarea, frecvența și secvențele de vârf, făcându-le potrivite pentru sisteme dinamice (precum semnalele senzoriale sau fluctuațiile de putere). Mai mult, unitățile de procesare se declanșează numai atunci când este necesar (controlat de evenimente), ceea ce la face ideale pentru hardware de putere redusă (cipuri neuromorfe precum Intel Loihi sau SpiNNaker). În momentul de față, RNS imită cel mai bine mecanismele reale ale creierului, ceea ce la face bune pentru modelarea proceselor cognitive sau simularea comportamentului neuronal. De asemenea, RNS-urile pot codifica informații atât în viteză, cât și în timp, crescând capacitatea de reprezentare a datelor, ele putând fi implicate în calcule complexe. Compatibilitatea lor cu hardware-ul neuromorf precum camerele DVS (Dynamic Vision Sensor), le face ideale pentru aplicații în robotică în timp real, viziune și sisteme auditive.

Dezavantajele RNS sunt legate de complexitatea procesului de învățare, durata procesului de antrenare, baza limitată de algoritmi și funcții accesibile (comparativ cu alte RNA), lipsa unui cadru de standardizare prin care să se poată verifica corect eficiența lor, și provocările care apar în implementarea lor, întrucât datele de intrare trebuie transformate în impulsuri / semnale de vârf.

### 8.6.Rețele hibride

Rețelele neuronale hibride prezentate aici au fost dezvoltate prin combinarea diferitelor tipuri de RNA și/sau algoritmi de învățare artificială. Aceste rețele

sunt: rețele covoluționale combinate cu rețele recurente, rețele multistrat antrenate prin învățarea în ansamblu, autocoder și clasificator clasic bazat pe învățarea automată, rețea standard împreună cu algoritmi de selecție a caracteristicilor, rețea standard antrenată prin învățarea prin întărire, generalizarea stivuită cu RNA, transfer learning (RNA ca bază, iar învățarea automată ca clasificator), și RNA plus algoritmi de grupare.

Dintre aceste opt rețele hibride, patru sunt folosite pentru a dezvolta aplicații dedicate rezolvării problemelor din sistemele electroenergetice: rețea multistrat antrenată prin învățarea de ansamblu, rețea standard împreună cu algoritmi de selecție a caracteristicilor, Autoencoder și clasificator clasic (multistrat perceptron), și rețea convoluțională combinată cu rețea recurentă. Aceste rețele hibride vor fi detaliate în continuare.

O rețea multistrat perceptron antrenată printr-un algoritm de învățare în ansamblu comparativ cu o rețea multistrat perceptron standard este mai robustă, întrucât nu este atât de sensibilă la semnalele zgomot, oferă o generalizare mai bună și acuratețe îmbunătățită datorită sinergiei de ansamblu. Avantajele menționate anterior rezultă din faptul că acest tip de rețele presupun combinarea mai multor rețele multistrat prin vot, mediere sau impuls și mai multe modele antrenate independent sau secvențial în funcție de metodologia de ansamblu abordată. Însă, complexitatea mai ridicată a acestor rețele cauzează o interpretabilitate și urmărire mai redusă a rezultatelor, plus un cost ridicat de calcul a procesului de antrenare.

Cele mai comune technici de asamblare a rețelelor multistrat perceptron într-o nuoă rețea hibridă sunt ambalarea (agregarea Bootstrap – antrenarea rețelelor pe diferite subseturi de date), boosting (exemplu – AdaBoost – antrenarea secvențială a rețelelor) și stivuirea (ieșirile de la mai multe rețele sunt utilizate ca intrare la un model de nivel superior (meta-învățator)).

Ambalarea prin agregarea Bootstrap presupune antrenarea mai multor rețele multistrat perceptron folosind subseturi aleatorii ale setului de antrenament. La final, rezultatul va fi obținut prin combinarea datelor de ieșire a rețelelor prin vot (votul majoritar) sau mediere (regresie). Matematica din spatele acestor metode de combinare a ieșirilor este descrisă prin (8.57) și (8.58).

Pentru votul majoritar,

$$\hat{y} = \arg max \sum_{m=1}^{M} \theta(f_m(x) = c). \tag{8.57}$$

și pentru regresie

$$\hat{y} = \frac{1}{M} \sum_{m=1}^{M} f_m(x). \tag{8.58}$$

Unde $\theta$ este funcția indicator, *c* variază între etichetele claselor, *M* este numărul de modele multistrat perceptron, $f_m(x)$ este un model de multistrat perceptron antrenat pe un subset de date $D^{(m)}$, iar $D = \{(x_i, y_i)\}, i = 1, N$ este baza de date perechi (date intrare-ieșire) folosită pentru antrenare și împărțită în subseturi.

Boosting-ul precum AdaBoost se realizează prin antrenarea secvențială a rețelelor multistrat implicate, iar fiecare nouă rețea se va focaliza pe greșelile rețelei precedente. Algoritmul AdaBoost pentru clasificarea binară începe prin inițializarea ponderilor primei rețele utilizate astfel: $w_i^{(1)} = \frac{1}{N}$. În continuare, pentru fiecare rețea *m* (*m*=1...*M*) se realizează pașii:

1. Antrenarea rețelei multistrat perceptron $f_m(x)$, prin folosirea poderilor $w_i^{(m)}$;
2. Determinarea erorii ponderilor

$$\epsilon_m = \sum_{i=1}^{N} w_i^{(m)} \theta(f_m(x_i) \neq y_i). \tag{8.59}$$

3. Calcularea modelului ponderilor

$$\alpha_m = \frac{1}{2} \ln \left(\frac{1-\epsilon_m}{\epsilon_m}\right). \tag{8.60}$$

4. Ajustarea ponderilor eșation

$$w_i^{(m+1)} = w_i^{(m)} \exp(-\alpha_m y_i f_m(x_i)). \tag{8.61}$$

   Normalizarea ponderilor $w_i^{(m+1)} \leftarrow \frac{w_i^{(m+1)}}{\sum w_i^{(m+1)}}$

5. Predicția finală

$$\hat{y} = sign(\sum_{m=1}^{M} \alpha_m f_m(x)). \tag{8.62}$$

Întrucât algoritmul lucrează cu date binare, datele de ieșire sunt adesea rescalate corespunzător pentru a obține date reale.

Stivuirea presupune antrenarea mai multor rețele multistrat perceptron și utilizarea ieșirilor rezultate de la ele pentru a antrena o rețea denumită meta-învățător (meta-learner). Astfel, pentru început sunt antrenate rețelele multistrat implicate ($f_m(x), m = 1, M$). În următoarea etapă se crează o nouă bază de seturi de antrenament din predicțiile rezultate de la aceste rețele, $D' = \{(z_i, y_i)\}, i = 1, N$, unde $z_i = [f_m(x), m = 1, M]$. Această nouă rețea (multistrat perceptron / regresie logistică / support vector machine) va fi antrenată pentru a prezice $y_i$ folosind datele intermediare $z_i$. În final, rezultatul prezis este

$$\hat{y} = g(f_m(x)) \tag{8.63}$$

O comparație între aceste trei metode de asamblare a rețelelor multistrat percetron arată că:

- Ambalarea este cea mai bună opțiune atunci când se urmărește robustețe și simplitate, în special cu date provenite de la senzori afectate de zgmote sau forme de undă în timp real.
- Boosting-ul este o metodă foarte bună dacă se lucrează cu seturi de date dezechilibrate sau dificile, dar trebuie să se depună atenție specială pentru a se evita supraadaptarea ponderilor la zgomot.
- Stivuirea dă cele mai bune rezultate în cazul sistemelor mari, modulare, în care mai multe tipuri de modele (inclusiv multistrat perceptron) captează diferite aspecte și o meta-rețea combină cunoștințele lor.

O rețea multistrat perceptron împreună cu un algoritm de selecție a caracteristicilor determină o rețea hibridă îmbunătățită datorită unei pre-etape în care datele de intrare sunt transformate sau selectate cu ajutorul unui algoritm de învățare automată. Scopul este de a crește acuratețea predicțiilor, a scădea timpul de antrenare a rețelei și a crește capacitatea de generalizare a rețelei hibride. Față de rețelele clasice, rețelelor hibride folosesc doar trăsăturile selectate sau de interes a datelor de intrare; dimensiunea lor este mai mică, fiind optimizată prin selecție; nu sunt sensibile la zgomote, întrucât acestea vor fi eliminate înainte de antrenarea rețelei; au timpul de antrenare mai mic datorită faptului că au mai puțini neuroni; au acuratețea mai ridicată deoarece se focalizează pe datele de intrare; determină rezultate mai ușor de interpretat.

Cele mai utilizate tehnici de selecție a datelor de intrare folosite în aplicațiile dedicate sistemelor electroenergetice sunt PCA (Principal Component Analysis), RFE (Recursive Feature Elimination) și Mutual Information (MI).

PCA are drept scop transformarea caracteristicilor originale ale datelor de intrare într-un nou set de date de componente ortogonale, care sunt ordonate după variație. Astfel, o bază de date de antrenament $X$, care este caracterizată de $n$ eșantioane și $d$ trăsături (fiecare eșantion are $d$ trăsături), va fi transformată astfel: se determină centrul datelor (8.64), apoi matricea de covarianță (8.65), se rezolvă problema cu valori proprii (8.66), se selectează primele $k$ componente, care au valoarea cea mai mare a vectorului propriu (8.67), iar la final se transformă datele (8.68).

$$\tilde{X} = X - \mu \tag{8.64}$$

$$C = \frac{1}{n}\tilde{X}^T\tilde{X} \tag{8.65}$$

$$Cv_k = \lambda_k v_k \tag{8.66}$$

$$V_k = [v_1, v_2, \dots, v_k] \tag{8.67}$$

$$Z = \tilde{X} V_k \tag{8.68}$$

Unde $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$, $\lambda_k$ și $v_k$ sunt prechile de valorea și vectorul proprii.

Această reducere a dimensiunii datelor de intrare păstrează caracteristicile de variație a datelor.

RFE va elimina caracteristicile fără importanță a datelor de intrare, prin folosirea ponderilor rețelei. Astfel, dacă un perceptron multistrat $f(x) = \sigma(Wx + b)$, cu ponderile $w_j$ asociate unei trăsături de intrare $x_j$ prin implicarea acestui algoritm se vor realiza următoarele activități: (1) antrenarea rețelei, (2) calcularea scorului de importanță a fiecărui trăsătură a intrărilor (8.69), (3) eliminarea trăsăturilor cu cel mai mic scor, repetarea primilor doi pași până se obțin numărul de trăsături dorite. În cazul în care modelele neuronilor sunt neliniare, importanța poate fi estimată prin folosirea derivatelor sau analiza perturbației.

$$score_j = \left|w_j\right|^2 \tag{8.69}$$

MI este o metodă de selecție prin care se măsoară cât de multă informație o anumită trăsătură *X*, deține despre ieșirea *Y*. Informația mutuală dintre două variabile aleatoare *X* și *Y* se determină astfel:

$$I(X;Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y) log\left(\frac{p(x,y)}{p(x)p(y)}\right) \tag{8.70}$$

$$I(X;Y) = \iint p(x,y) log\left(\frac{p(x,y)}{p(x)p(y)}\right) dxdy \tag{8.71}$$

Unde $p(x,y)$ funcția de densitate de probabilitate comună, $p(x)p(y)$ sunt distribuțiile marginale.

Aceste trei metode de selecție sau transformare a datelor ajută la reducerea complexității RNA (RFE), captează variația la nivel de sistem în datele cu dimensiuni mari (PCA), respectiv dezvăluie influențe neliniare ascunse (MI).

Combinația dintre un autoencoder (codificator automat) și o rețea perceptron multistrat determină o rețea hibridă, care într-o primă etapă realizează extragerea nesupravegheată a caracteristicilor, iar apoi perceptronul antrenat va realiza o reprezentare latentă comprimată. Astfel, codificatorul automat reduce datele de intrare la caracteristicile sale esențiale, iar RNA învață să clasifice aceste

caracteristici cu etichetele de ieșire. Comparativ cu un simplu multistrat perceptron, rețeaua hibridă are la intrare date cu trăsături latent comprimate, învață să reducă dimensiunea și zgomotele din datele de intrare, are rezultate ușor interpretabile dacă codificatorul este bine structurat, oferă acuratețe îmbunătățită cu ajutorul unui codificator bine antrenat, nu suferă de supraadaptarea ponderilor; are antrenarea atât supravegheată cât și nesupravegheată.

Baza matematică a acestei rețele hibride este definită prin (8.72)-(8.75) pentru codificatorul automat (codificator-decodificator), (8.76) pentru RNA. Astfel, codificatorul comprimă datele de intrare *x* în codul latent *z*.

$$z = f_{cod}(x) = \sigma(W_c x + b_c) \tag{8.72}$$

Apoi se reconstruiește intrarea din codul latent

$$\hat{x} = f_{dec}(z) = \sigma(W_d z + b_d) \tag{8.73}$$

Unde $W_c$ și $W_d$ sunt ponderile codificatorului automat, iar $\sigma$ este funcția de activare.

La următorul pas se urmărește minimizarea pierderilor de reconstrucție

$$L_{AE} = \|x - \hat{x}\|^2 = \|x - f_{dec}(f_{cod}(x))\|^2 \tag{8.74}$$

Odată antrenat, codificatorul automat este abandonat și se folosește

$$z = f_{cod}(x) \tag{8.75}$$

ca date de intrare pentru perceptronul multistrat, care va oferi la ieșire probabilitățile claselor.

$$\hat{y} = f_{cfl}(z) = softmax(W_c x + b_c) \tag{8.76}$$

Unde $W_c$ și $b_c$ sunt ponderile și valorile bias, $\hat{y}$ – valoarea prezisă a probabilității clasei.

Aceste rețele hibride sunt mai rapide și prezintă o acuratețe mai mare a rezultatelor mari întrucât codificatorul automat reduce mult reprezentarea datelor de dimensiune mare.

Combinația dintre o rețea convoluțională împreună cu o rețea recurentă rezultă într-o rețea hibridă puternică, care este foarte folositoare în analiza semnalelor spațiale și temporale, deci utilă în aplicațiile dedicate sistemelor electroenergetice. În cadru rețelei hibride, rețeaua convoluțională acționează ca un extractor de caracteristici, iar rețeaua recurentă învață cum evoluează aceste caracteristici de-a lungul timpului sau al pașilor secvențiali. Arhitectura rețelei

hibride cuprinde stratul de intrare care preia datele de intrare, urmând straturile rețelei convoluționale, straturile rețelei recurente și stratul de ieșire care oferă clasificarea sau regresia. O comparație între o rețea convoluțională, o rețea recurentă și una hibridă, arată că rețeaua hibridă are în plus următoarele avantaje: capacitatea de a manipula atât date temporale cât și spațiale și capacitatea de a determina caracteristicile datelor de intrare este excelentă. Principalul dezavantaj al acestui tip de rețele este timpul mare de antrenare.

Etapele detaliate în utilizarea acestui tip de rețea hibridă sunt:

1. Citirea datelor de intrare sub forma unor vectori de serii de timp multivariate.

   $X = [x_1, x_2, \dots, x_T]$. $T$ – numărul de etape de timp, $n$ – numărul de caracteristici.

2. Extragerea caracteristicilor de către componenta convoluțională

$$z_t^{(j)} = \sigma\left(\sum_j w_i^{(j)} * x_t^{(i)} + b^{(j)}\right). \tag{8.77}$$

   Unde * este operația de convoluție, $w_i^{(j)}$ – ponderile centrelor, $z_t^{(j)}$ - harta caracteristicilor de ieșire la pasul de timp $t$ pentru filtrul $j$, $\sigma$ – funcția de activare.

3. Modelarea temporală folosind rețeaua recurentă

Pentru o rețea recurentă clasică

$$h_t = tanh(W_h h_{t-1} + W_z z_t + b_h). \tag{8.78}$$

Pentru o rețea memorie pe termen lung

Poarta uită - $f_t = \sigma\left(W_f z_t + U_f h_{t-1} + b_f\right)$. (8.79)

Poarta de intrare - $i_t = \sigma(W_i z_t + U_i h_{t-1} + b_i)$

Poarta de ieșire – $o_t = \sigma(W_o z_t + U_o h_{t-1} + b_o)$

Starea celulei – $c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$

$$\tilde{c}_t = \tanh(W_c z_t + U_c h_{t-1} + b_c).$$

Starea ascunsă - $h_t = o_t * \tanh(c_t)$.

Unde $W$, $U$ și $b$ sunt ponderile și valorile bias, iar * operația de multiplicare în funcție de element.

4. Startul de ieșire va determina valoarea de ieșire în funcție de scopul rețelei
   Pentru clasificare

$$\hat{y} = softmax(W_o h_T + b_o) \tag{8.80}$$

   Pentru regresie

$$\hat{y} = W_o h_T + b_o \tag{8.81}$$

5. Funcția de pierderi arată deviația valorii prezise față de valoarea reală.
   Pentru clasificare – cross-entropy

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{8.82}$$

   Pentru regresie MSE

$$L = \frac{1}{N}\sum_{i=1}^{N} \|y_i - \hat{y}_i\|^2 \tag{8.83}$$

Această rețea hibridă are cea mai complexă arhitectură, având capabilitățile celor două rețele din care este formată, fără a avea slăbiciunile lor.

Surse bibliografice din care se poate aprofunda tema rețelelor neuronale artificiale sunt [36] – [49].

## 8.7.Întrebări și răspunsuri

1. Rețelele neuronale artificiale avansate sunt total diferite de rețelele neuronale tradiționale.
   Adevărat / Fals

2. Rețelele neuronale recurente se caracterizează prin faptul că informația circulă atât de la intrare spre ieșire, cât și invers.
   Adevărat / Fals

3. Rețelele neuronale artificiale care se apropie cel mai mult rețelele neuronale biologice sunt rețele spiking.
   Adevărat / Fals

4. Rețelele hibride sunt obținute prin combinația rețelelor neuronale tradiționale între ele sau cu alți algoritmi de învățare automată.

Adevărat / Fals

5. Rețelele neuronale artificiale pot fi folosite pentru a rezolva orice problemă din sistemele electroenergetice.
   Adevărat / Fals

# 9

# Rețele neuronale artificiale
## Aplicații în managementul energiei

Prezentul capitol se focalizează pe prezentarea aplicațiilor care folosesc rețele neuronale artificiale pentru rezolvarea problemelor caracteristice sistemelor electroenergetice, și în special a managementului energiei. În plus, se descriu metodele folosite pentru implementarea rețelelor neuronale artificiale. La finalul capitolului sunt prezentate studii publicate în literatura de specialitate, care au drept subiect utilizarea calculului neuronal pentru a soluționa probleme din managementul energiei electrice, dar distribuite pe cele patru categorii, și anume în producerea, transportul, distribuția și utilizarea energiei electrice.

### 9.1.Aspecte generale

Rețelele neuronale artificiale sunt tehnicile de IA care în momentul de față au cele mai multe aplicații, în cele mai diverse domenii de activitate umană. Acest fapt se datorează principalelor capabilități ale RNA, și anume capacitatea de a învăța, de a generaliza și a sintetiza. Astfel, RNA sunt folosite pentru:

- Clasificare: dacă se furnizează reţelei neuronale un model de intrare, la ieşire se obţine clasa sau categoria căruia îi aparţine modelul respectiv.
- Asocierea modelelor: dacă se formulează reţelei neuronale un model de intrare, la ieşire se obţine un model asociat primului pe baza unor reguli transmise reţelei în etapa de învăţare.
- Completarea modelelor: dacă se prezintă reţelei neronale un model de intrare incomplet (din care lipsesc o serie de informaţii), la ieşire se obţine modelul reconstituit.
- Filtrarea semnalelor: la intrarea reţelei se aplică un semnal afectat de „zgomot”, iar la ieşire se obţine un semnal filtrat din care s-a eliminat total sau parţial componenta de „zgomot”.

- Optimizare: dacă se reprezintă reţelei neuronale un model de intrare care reprezintă valorile iniţiale ale unei probleme de optimizare, la ieşire se obţine un set de variabile, care reprezintă soluţia optimă sau suboptimă a problemei.
- Control: dacă modelul de intrare care se transmite reţelei reprezintă starea curentă a unui element de control şi răspunsul care se doreşte de la acesta, atunci la ieşire se obţine secvenţa de comenzi care conduce la acest răspuns.

În sistemele electroenergetice, rețelele neuronale artificiale sunt implicate în rezolvarea problemelor:

1. Producerea energiei electrice
    - Prognoza producerii energiei electrice a unităților de generare din surse regenerabile.
    - Producerea optimă de energie – optimizarea multi-obiectiv și optimizarea producerii combinate căldură-electricitate.
    - Modelarea și controlul generatoarelor – sisteme de control a generatoarelor, modelarea dinamicii generatoarelor neliniare și controlul sistemului de excitator-generator.
    - Sisteme de energie regenerabilă – predicția vitezei vântului și a iradierii solare, urmărirea punctului de putere maximă (MPPT) pentru panourile fotovoltaice și predicția defecțiunilor la turbinele eoliene.
    - Monitorizarea stării și diagnosticarea defecțiunilor - detectarea defecțiunilor mecanice/electrice la turbine și generatoare și analiza vibrațiilor.
2. Transportul energiei electrice
    - Estimarea circulației de puteri și a stării sistemului - rezolvarea problemelor de circulație de putere folosind aproximatori bazați pe RNA și estimarea stării în timp real cu date incomplete sau afectate de zgomote.
    - Detectarea și clasificarea defecțiunilor – locația și clasificarea tipului de defecțiune pe liniile de transport a energiei electrice și sisteme inteligente de protecție pe bază de clasificatoare RNA.
    - Analiza stabilității – predicția stabilității tranzitorie și predicția limitei de stabilitate a tensiunii.

- Sisteme de control – îmbunătățirea monitorizării și controlului zonei extinse și evaluarea dinamică a securității folosind RNA recurente și hibride.
- Estimarea parametrilor de linie, adică a impedanței, rezistenței și reactanței liniei de transmisie din date în timp real.

3. Distribuția energiei electrice
   - Detectarea defecțiunilor și restabilirea alimentării – localizarea defecțiunii, izolarea și restaurarea utilizând RNA, predicția defecțiunilor componentelor la transformatoarele de distribuție și sisteme cu refacere automată folosind sisteme inteligente.
   - Reglarea tensiunii și reducerea pierderilor – plasarea optimă a condensatoarelor, controlul în timp real al U/Q și controlul transformatoarelor prin schimbarea ploturilor.
   - Prognoza pe termen scurt a nivelului de încărcare a transformatoarelor de distribuție.
   - Monitorizarea calității energiei electrice – detectarea și clasificarea distorsiunilor armonice, a flickerului, a golurilor de tensiune, respectiv supratensiunilor.
   - Integrarea generatoarelor distribuite – gestiunea circulației bidirecționale de putere și sisteme de management al energiei.
4. Utilizarea energiei electrice
   - Managementul inteligent al energiei – optimizarea răspunsului la cererea de putere, sisteme de management al energiei la consumatorii casnici și dezagregarea sarcinii.
   - Modelarea consumului de energie electrică – modelarea comportamentului consumatorilor pentru planificarea energetică și modele de consum dinamic bazate pe RNA.
   - Eficiență energetică și prognoză – prognoza consumului de energie în clădiri și spații industriale, șiprognoza și controlul utilizării energiei electrice la nivel de echipament.
   - Detectarea anomaliilor și securitatea cibernetică – detectarea curbelor de sarcină anormale și detectara intruziunilor în contoarele inteligente și sistemele energetice bazate pe IoT.
   - Vehicule electrice – prognoza consumului vehiculelor și controlul acestora, dar și controlul schimburilor de putere între vehicule și rețea.

Particularizând aceste aplicații pentru rezolvarea problemelor asociate managementului energie electrice, vor face referire la aspectele care au legătură cu problemele de optimizare, control, prognoză, și strategii de creștere a eficienței energetice.

Dacă luăm în considerare tipurile de RNA folosite în rezolvarea problemelor de management al energiei electrice, atunci rezultă informațiile din tabelul 9.1.

**Tabelul 9.1.** Utilizarea RNA în managementul energiei

| **Tipul RNA** | **Aplicații** |
|---|---|
| Rețele feedforward / multistrat perceptron | Prognoza consumului de energie (pe termen scurt, mediu și lung).<br>Predicția generării de energie regenerabilă (de exemplu, iradierea solară, viteza vântului).<br>Predicția consumului de energie pentru clădiri și industrie.<br>Dispecerare economică a consumului pentru condiții statice simple.<br>Clasificarea profilului de încărcare și gruparea consumatorilor.<br>Estimarea pierderilor de putere în rețelele de distribuție. |
| Rețele recurente | Predicția în serie de timp a consumului și producerii de energie cu dependențe temporale.<br>Modelarea dinamică a răspunsului la prețul energiei în managementul cererii.<br>Predicția modelului de încărcare a vehiculelor electrice bazată pe comportamentul utilizatorului.<br>Prognoza consumului de energie unde valorile trecute le influențează puternic pe cele viitoare. |
| Rețele memorie pe termen lung | Prognoza de consumului foarte precisă, cu dependențe pe termen lung.<br>Predicția producției de energie regenerabilă pe diferite intervale de timp.<br>Modelarea comportamentului de stocare a bateriei în sistemele de management al energiei. |

| | |
|---|---|
| | Învățarea modelelor de consum de energie pentru microrețele. |
| Rețele convoluționale | Prognoza multivariată a energiei pentru a extrage caracteristici spațiale/temporale.<br>Detectarea anomaliilor în modelele de consum de energie.<br>Clasificarea consumului de energie în clădiri inteligente sau în aparate. |
| Rețele hibride | Gestionarea cererii în condiții incerte.<br>Programarea optimă a generatoarelor distribuite și stocării.<br>Managementul multi-obiectiv al energiei.<br>Control inteligent în timp real al energiei în microrețele cu surse regenerabile. |
| Alte rețele neuronale artificiale | Sisteme de management al energiei la scară largă în care modele diferite gestionează subsisteme separate.<br>Controlul ierarhic al rețelelor inteligente și al microrețelelor.<br>Prognoza și controlul sarcinii specifice regiunii.<br>Prognoza sarcinii pe termen scurt cu viteză mare și generalizare.<br>Controlul tensiunii/puterii reactive în timp real în managementul distribuției.<br>Control la scară mică a generatoarelor distribuite în microrețele. |

Aplicațiile RNA dedicate rezolvării problemelor din managementul energiei (dar nu numai) sunt implementate sub formă de aplicații informatice – software, sau au la bază dispozitive electronice dedicate. Astfel, în continuare sunt prezentate aspectele ce țin de implementarea rețelelor neuronale artificiale în subcapitolul următor.

### 9.2.Implementarea RNA

Rețelele neuronale artificiale pot fi implementate atât ca software, cât și ca hardware, în funcție de aplicație, cerințele de viteză și constrângeri precum puterea sau costul.

Implementarea sub formă de software este abordarea standard de dezvoltare și utilizare a RNA. Acestea sunt dezvoltate pe sisteme electronice dedicate precum calculatoare computer / PC, servere, platforme cloud și sisteme încorporate. Instrumentele informatice și bibliotecile dedicate folosite în prezent pentru dezvoltarea de RNA software sunt Phyton (bibliotecile - TensorFlow, PyTorch, Keras), MATLAB (unealtă dedicată și linii de cod personalizate), C/C++ (implementări încorporate) și Java/Scala (în unele platforme industriale). Câteva exemple de astfel de aplicații sunt dedicate pentru: prognoza, detectarea defecțiunilor, analiza semnalului și sisteme de control.

Phyton și librăriile bazate pe Phyton sunt cele mai utilizate datorită ușurinței sale în utilizare, comunității active de specialiști și bibliotecilor complete care mereu sunt actualizate, întrucât Phyton este un limbaj de programare gratuit (open-source), precum și bibliotecile sale (cu unele excepții). Cele mai folosite biblioteci sunt TensorFlow, PyTorch, Keras, Scikit-learn și JAX. TensorFlow este dezvoltat de Google și suportă atât operațiuni de nivel scăzut, cât și aplicații de nivel înalt (prin Keras). Această bibliotecă de funcții este optimizată pentru utilizarea CPU (Central Processing Unit), GPU (Graphic Processing Unit) și TPU (Poliuretan termoplastic), astfel ea este potrivită atât în cercetare, cât și în producție. PyTorch este dezvoltat de Meta (Facebook) și folosit în special în cercetarea academică. Această bibliotecă prezintă grafice de calcul dinamic, astfel este ideală pentru dezvoltarea rapidă de prototipuri și rețelele neuronale artificiale experimentale. Keras este o bibliotecă cu funcții și proceduri informatice de nivel înalt, care a fost inițial dezvoltată ca un proiect independent. Prin implicarea acestei biblioteci, procesul de construire, dezvoltare și instruire a unei RNA este mult simplificat. Aceasta este acum integrată în TensorFlow. Scikit-learn este o librărie proiectată pentru învățarea automată tradițională, dar include modele de bază ale RNA. Astfel, aceasta este foarte utilă pentru dezvoltarea de RNA de tipul perceptron multistrat, și preprocesare. JAX este o bibliotecă dezvoltată de Google, și se concentrează pe calculul numeric de înaltă performanță și diferențierea automată. Această bibliotecă este cel mai des folosită în cercetarea arhitecturilor experimentale a RNA.

Unealta Deep Learning Toolbox, DLT de la MATLAB oferă funcții încorporate pentru proiectarea, antrenamentul și vizualizarea RNA, precum și o interfață grafică dedicată pentru dezvoltarea rețelelor. Mai mult, prin folosirea DLT, utilizatorul are posibilitatea integrării aplicației cu Simulink pentru aplicații de control și procesare a semnalului. Un alt avantaj în utilizarea DLT este suportul pentru generarea de cod pentru sistemele încorporate.

Limbajele C/C++ și Java sunt utilizate în aplicațiile de sisteme încorporate (embedded), și în situațiile unde performanța este un aspect esențial. Biblioteci care au la bază sunt FANN (Rețea Neuronală Artificială Rapidă), care este o bibliotecă catalogată drept C ușoară, care este potrivită pentru aplicații cu resurse reduse. O altă bibliotecă care are la bază C++ este minuscul-dnn, care este excelentă pentru aplicații de rețele neuronale mici, inclusiv sisteme încorporate. DL4J (DeepLearning4J) este un cadru / framework dedicat dezvoltării de aplicații de învățare profundă bazat pe Java. Un avantaj a acestei unelte informatice este faptul că se poate integra cu Apache Spark pentru instruire distribuită la scară largă.

RNA software sunt cele mai populare aplicații ale calculului neuronal deoarece, aplicațiile informatice sunt simplu de dezvoltat și actualizat, dar și flexibile, astfel modelele software sunt ușor de antrenat, modificat și testat. Mai mult, se poat face experimente, încerca diferite arhitecturi foarte rapid. De aceea, RNA software sunt folosite în etapa de prototip a unei aplicații, mai mult ele sunt ideale pentru studiile de cercetare, dezvoltare și învățământ. RNA software au dedicate instrumente informatice puternice, precum și platforme independente care pot fi folosite cu diferite sisteme informatice (PC, cloud, Linux, încorporate). Dezavantajele folosirii acestor RNA software rezidă din necesarul ridicat de resurse energetice, dependența de resurse externe, precum și faptul că ele sunt lente în cazul aplicațiilor în timp real.

RNA hardware sunt mai puțin folosite în prezent, dar numărul lor este în creștere. În această abordare, RNA sunt implementate fizic pentru sisteme ultra-rapide, cu putere redusă sau în timp real. Aceste sisteme electronice sunt hardware de tip digital, precum FPGA, ASIC, microcontrolere, hardware de tip analogic (circuite concepute pentru a imita comportamentul neuronilor), și cipurile neuromorfe (cipuri proiectate ca creierul uman). Trei exemple de RNA hardware sunt protecția în timp real a sistemelor generatoare, contoare inteligente și dispozitive edge (senzori IoT, detectoare de erori).

FPGA-uri (Field Programmable Gate Arrays) ca circuite integrate reconfigurabile sunt un veritabil suport pentru calculul paralel, ceea ce la face ideale pentru inferența și instruirea RNA. Exemple de astfel de instrumente electronice sunt cele produse de Xilinx Vivado, Intel Quartus, HLS (High-Level Synthesis). Ele sunt populare pentru aplicații care au legătură cu sisteme de alimentare, procesarea semnalului și controlul în timp real. Punctele forte în utilizarea FPGA-urilor în aplicații RNA reies din faptul că acestea sunt reconfigurabile, realizează procesare paralelă rapidă, și au latență scăzută. Dar,

au un design complex, necesită expertiză hardware (HDL/VHDL/Verilog sau HLS/C), ceea ce nu le fac să nu fie deschise unui grup mare de ingineri.

ASIC-uri (Circuite integrate specifice aplicației) sunt cip-uri personalizate construite special pentru a implementa o RNA fixă. În consecință, produsul rezultat este extrem de rapid și eficient din punct de vedere energetic; acesta fiind principalul motiv pentru care este folosit în industrie pentru produse comerciale (de exemplu, Google TPU v1 este bazat pe ASIC). Avantajele utilizării acestor cipuri sunt viteza mare de lucru și consumul foarte scăzut, dar proiectarea lor este costisitoare și necesită timp, mai mult nu sunt reprogramabile.

Microcontrolere și DSP (procesoare de semnal digital) sunt procesoare încorporate care pot rula modele RNA la scară mică. Ele sunt potrivite pentru dezvoltarea de aplicații precum senzori inteligenți, controlul motorului sau aplicații care necesită cu putere redusă. Cele mai populare microcontrolere sunt STM32, ESP32, Arduino, și TI C2000. Abordarea în utilizarea microcontrolerelor este următoarea: modelele RNA sunt de obicei pre-antrenate și implementate folosind aritmetica în virgulă fixă, adică rețelele sunt antrenate înainte de a fi implementate (antrenarea nu se realizează pe microcontroler, ci în afara lui, el fiind doar un suport pentru RNA antrenată). În concluzie, avantajele utilizării acestor dispozitive sunt costul redus și integrarea ușoară cu senzori., iar dezavantajele sunt procesare și memorie limitate; ceea ce nu le face potrivite pentru RNA mari.

Prin utilizarea implementărilor analogice se încearcă să se imite fizic comportamentul neuronilor și al sinapselor folosind circuite electronice. Astfel, pentru dezvoltarea acestor RNA sunt utilizate amplificatoare operaționale, memristoare, condensatoare și tranzistoare pentru a reproduce comportamentul de tip neuron. Mai mult, ele pot fi construite cu componente discrete sau pe cipuri analogice personalizate. Aceste hardware RNA sunt foarte rapide și eficiente din punct de vedere energetic, datorită absenței logicii digitale. Deci, punctele forte ale acestor RNA sunt putere ultra-scăzută consumată și procesarea continuă în timp real, iar punctele slabe rezultă din faptul că dispozitivele sunt sensibile la zgomot, temperatură și variații de proces, dar și unt greu de scalat și programat.

Hardware neuromorfic sunt RNA fizice inspirate de creier, care folosesc neuroni de vârf (spiking) sau alte modele plauzibile din punct de vedere biologic. Cele mai folosite dispozitive din această categorie sunt cipurile neuromorfe, care pot fi proiectate pentru a simula structurile neuronale și dinamica lor. Avantajele acestor RNA sunt lucrul în timp real, eficient din punct de vedere energetic, acceptă calculul bazat pe evenimente. Dezavantajele lor rezultă din faptul că

acestea sunt în stadiul experimental, și necesită modele specifice, care nu sunt compatibile cu modelele standard de învățare profundă.

RNA bazate pe hardware oferă o mai mare viteză de procesare și de rezolvare a problemelor. Acesta este motivul pentru care ele sunt folosite cu precădere în aplicațiile industriale și tehnice. Mai mult, circuitele electronice folosite în RNA sunt mai eficiente energetic, compacte și de-sine-stătătoare, precum și fiabile în rezolvarea problemelor din sistemele critice. Totuși, RNA hardware sunt greu de actualizat (schimbarea structurii RNA sau a ponderilor poate necesita reprogramarea sau remodelarea dispozitivului), complicat de dezvoltat (inginerul are nevoie de cunoștințe profunde atât despre RNA, cât și a circuitelor digitale/analoge), au flexibilitate limitată (mai puțin adaptabilă la diferite probleme) și un cost inițial ridicat.

O sinteză a comparației dintre RNA software și hardware este prezentată în tabelul 9.2.

**Tabelul 9.2.** Comparație între RNA software și RNA hardware

| **Caracteristică** | **RNA software** | **RNA hardware** |
|---|---|---|
| Flexibilitate | Foarte mare | Limitată (rezultate mai bune FPGA) |
| Viteză | Depinde de resurse | Foarte rapidă |
| Eficiență energetică | Moderată | Mare |
| Timpul de dezvoltare | Rapid și ușor | Mai complex |
| Cost | Mic | Mare pentru dispositive dedicate |

### 9.3.Aplicații în producerea energiei electrice

Cele mai multe aplicații ale RNA în managementul energiei sunt pentru a realiza prognoza cantității de energie electrică produsă de generatoarele care folosesc surse energetice regenerabile, precum energia solară și energia eoliană. O astfel de aplicație este prezentată în articolul

[50] Alamin, Y.I.; Anaty, M.K.; Álvarez Hervás, J.D.; Bouziane, K.; Pérez García, M.; Yaagoubi, R.; Castilla, M.d.M.; Belkasmi, M.; Aggour, M. Very Short-Term Power Forecasting of High Concentrator Photovoltaic Power

Facility by Implementing Artificial Neural Network. Energies 2020, 13, 3493. https://doi.org/10.3390/en13133493,

Unde autorii propun utilizarea unei rețele neuronale artificiale Radial Basis Functions Neural Network (RBFNN) – rețea neuronală cu funcții de bază radială pentru a obține prognoza pe termen foarte scurt (câteva secunde până la o oră) a unei centrale potovoltaice cu concentrație mare.

Centrala fotovoltaică folosită în studiu este ilustrată în figuta 9.1.

**Fig. 9.1.** Centrala fotovoltaicu cu concentrație mare folosită în [50]

RNA de tipul BRF este compusă din trei straturi de neuroni: intrare, ascuns și ieșire, a căror neuroni sunt complet conectați între ei, precum la o rețea clasică feedforward. Fiecărei variabile de intrare i se atribuie un nod. Apoi, semnalele de intrare trec fără a fi afectate de ponderi către stratul ascuns. Stratul ascuns conține funcții de transfer, numite și RBF (radial basis function – funcție de bază radială). O funcție RBF are o poziție centrală și o rază. Cea mai mare valoare de ieșire este obținută atunci când variabilele de intrare sunt aproape de poziția centrală. Pe de altă parte, funcția scade constant atunci când distanța față de centru crește. Viteza de scădere a funcției RBF este definită de rază. Mai exact, când raza este

mică, scăderea va fi rapidă, în caz contrar, când raza este mare, scăderea va fi lentă.

Datele brute folosite în studiu au fost obținute din măsurători, care acoperă 92 de zile între anii 2016 și 2018. Aceste date au fost filtrate, fiind eliminate zgomotele șinerorile de măsurare, în final fiind disponibile 29960 de eșantioane, care au fost folosite pentru antrenament (35%), generalizare (35%) și testare (30%). Seturile de date folosite pentru cele trei acțiuni au fost distribuite aleator. Fig. 9.2 prezintă exemple de date folosite în studiu.

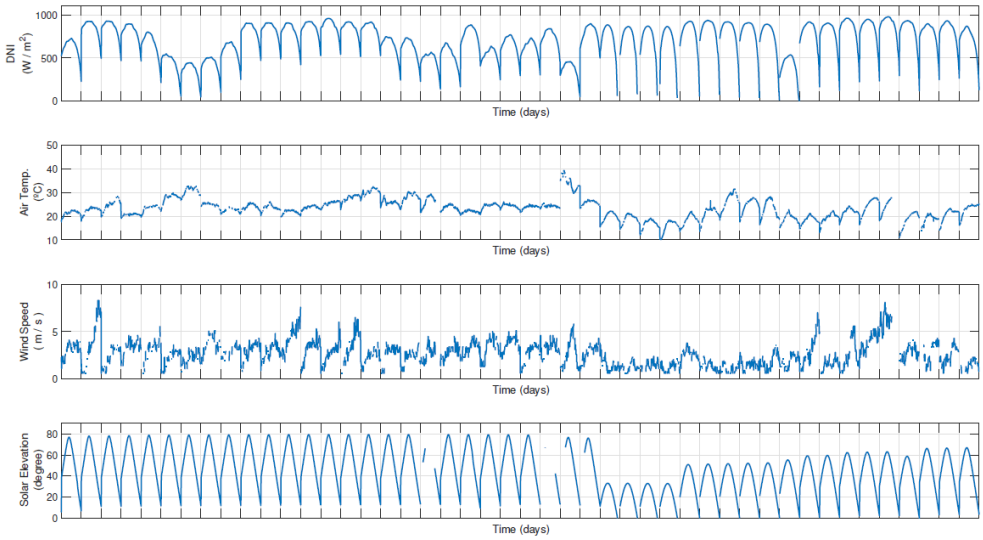



**Fig. 9.2.** Date folosite în studiu pentru antrenare, generalizare și testare [50]

În studiu, factorii care au fost luați în considerare ca influențând prognoza, și în consecință au devenit datele de intrare ale rețelei sunt direcția vântului - Wd, temperatura aerului -Tair, AM bazat pe altitudinea solară (măsurată), viteza vântului - Ws, unghiul de azimut al tracker-ului și iradierea normală directă - DNI, plus întârzierile de una și două ori ale puterii generatorului, care reprezintă semnalul de putere întârziat cu una și două ori.

Fig. 9.3. ilustrează structura general a rețelei, cu toate intrările sale externe și feedback-ul după întârzieri de una și două ori (întârzieri de una și două ori) de la puterea de ieșire.

În antrenarea (învățare supravegheată) și testarea rețelei pentru a calcula devierea mărimii prezise față de valorea reală a fost folosit indicele erorii medii pătratice (RMSE). Mai mult, au fost dezvoltate și testate mai multe modele ale

RNA, cu un număr diferit a nodurilor, dar modelele care au dat cele mai bune rezultate au avut 21, 18, respectiv 24 noduri.

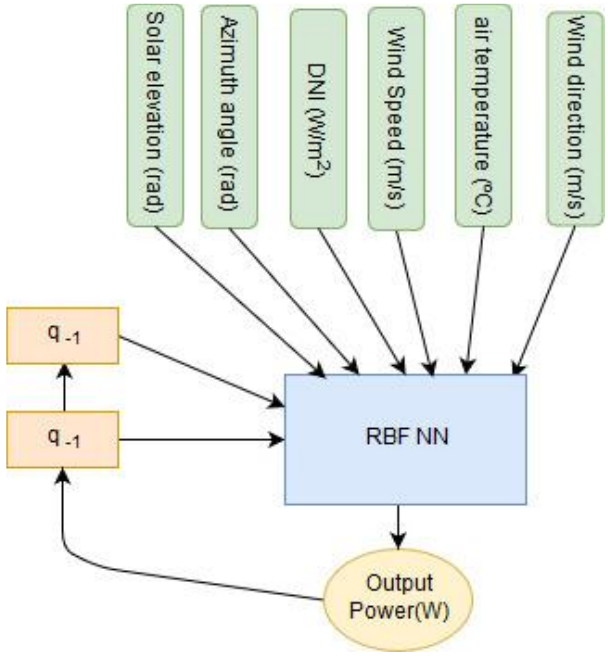



**Fig. 9.3.** Structura rețelei propuse în [50]

La final, autorii au concluzionat că acest tip de RNA se poate folosi pentru prognoza pe termen foarte scurt, dând rezultate bune, dar în cazul unor intervale mai mari de timp, va da erori, deoarece modelul neuronal se degradează.

RNA sunt folosite în managmenetul energiei și pentru predicția unor parametrii precum rata de încălzire (debitul termic) a unei centrale electrice cu ciclu combinat, care este prezentată în articolul

[51] Arferiandi, Y.D.; Caesarendra, W.; Nugraha, H. Heat Rate Prediction of Combined Cycle Power Plant Using an Artificial Neural Network (ANN) Method. Sensors 2021, 21, 1022. https://doi.org/10.3390/s21041022.

Debitul termic al unei centrale electrice cu ciclu combinat (CCPP) este utilizat pentru a evalua eficiența centralei. Predicția debitul termic are rolul de a sprijini personalul de mentenanță în monitorizarea eficienței centralei. Variabilele de intrare sunt aportul de căldură din gazul combustibil (P1), procentul de CO2 (P2) și puterea de ieșire (P3). Aproximativ 4322 de date brute sunt generate de sistemul de supraveghere digitală a centralei, date care au fost

utilizate pentru antrenamentul și testare. În studiu au fost dezvoltate șapte variații de parametri pentru a găsi cea mai bună variație a parametrilor pentru a prezice debitul termic cât mai corect. Rețeaua neuronală folosită este de tipul feef-forward cu propagarea înapoi a erorii și învățare supravegheată.

În studiu au fost considerate trei arhitecturi de RNAF, și anume cu un parametru, cu doi, respectiv cu trei parametrii. Structura celor trei rețele este ilustrată în fig. 9.4. În cazul în care s-a folosit un singur parametru, RNA a avut ca date de intrare câte un parametru pe rând (P1, P2 și P3), RNA cu doi parametrii a avut ca variabile de intrare doi câte doi din cei trei parametrii (P1+P2, P1+P3, P2+P3), iar cu trei parametrii P1+P2+P3.

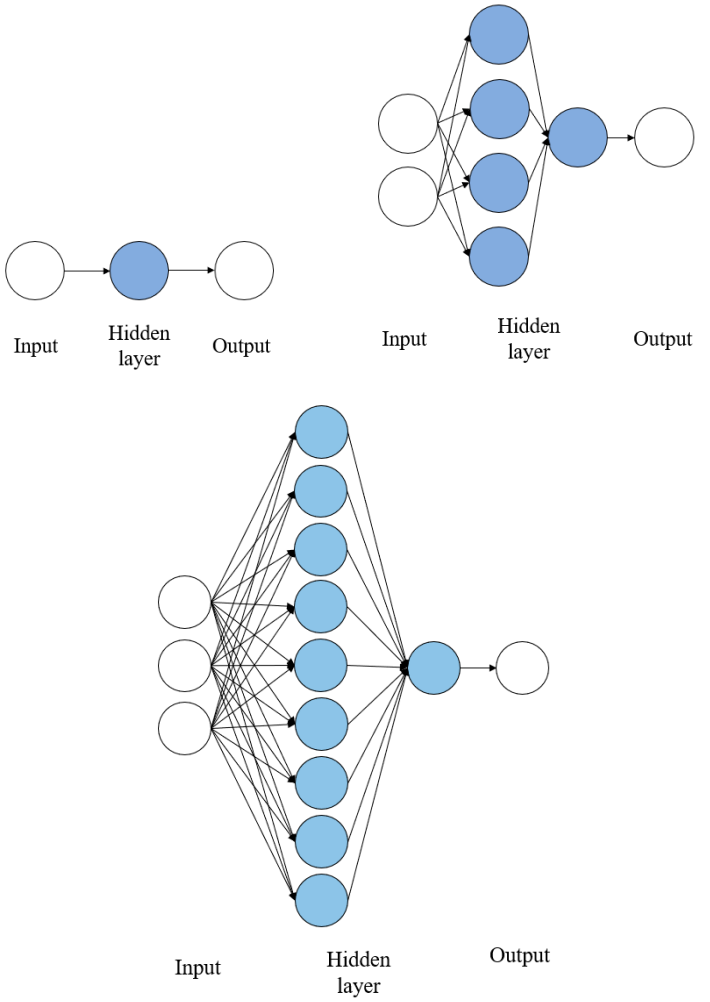



**Fig. 9.4.** Structura rețelelor folosite în [51]

Rețeaua a fost implementată, antrenată și testată prin utilizarea uneltei dedicate din MATLAB. Rezultatele obținute sunt în detaliu discutate în articol. Fig. 9.5 prezintă capturile de ecran introduse de autori în articol, din cele trei variații ale RNA.

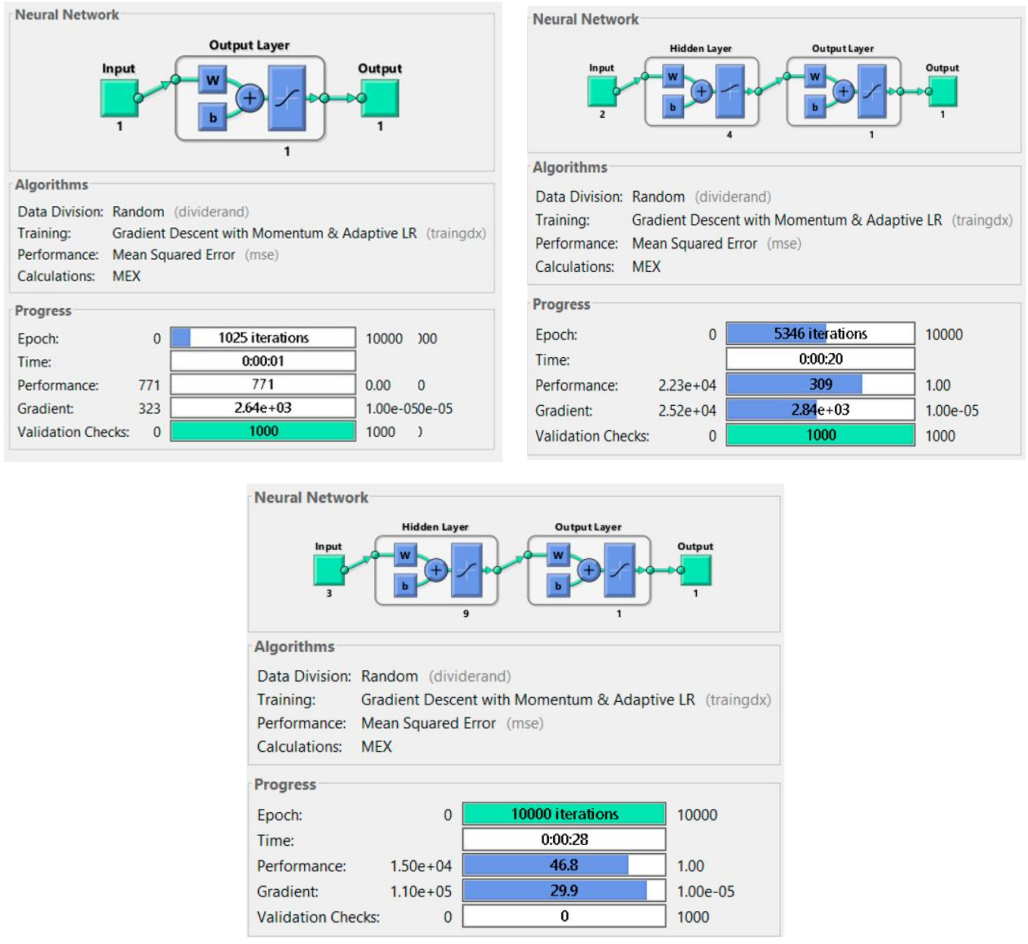



**Fig. 9.5.** Implementarea RNA în MATLAB [51]

În studiu, 75% din date au fost folosite pentru antrenare, iar 25% pentru testare. Pentru evaluarea rezultatelor prezise au fost folosit indicele $R^2$, care arată eroare rezultatului față de valoarea reală, iar acestea au arătat că folosirea împreună a celor trei parametrii dă cele mai bune rezultate cu o eroare medie de 2,52%.

### 9.4.Aplicații în transportul energiei electrice

Un alt exemplu de implicare a RNA în rezolvarea unor probleme din managementul energiei este articolul

[52] Al-Majidi, S.D.; Kh. AL-Nussairi, M.; Mohammed, A.J.; Dakhil, A.M.; Abbod, M.F.; Al-Raweshidy, H.S. Design of a Load Frequency Controller Based on an Optimal Neural Network. Energies 2022, 15, 6223. https://doi.org/10.3390/en15176223.

Unde autorii folosesc o rețea neuronală optimizată pentru a proiectarea unui regulator de frecvență.

Strategia de lucru folosită de autori în studiul prezentat în articolul menționat este ilustrată în fig. 9.6. Autorii au parcurs șase etape pentru a finaliza studiul: proiectarea regulatorului, colectarea datelor, procesarea datelor, antrenarea RNA, implementarea regulatorului optimizat prin RNA și procesarea și analiza rezultatelor.

Rețeaua neuronală este folosită pentru a obține răspunsul în frecvență a regulatorului în situația uni sistem electroenergetic caracterizat de o singură zonă (un singur generator împreună cu consumatorii corespunzători și consumatorii vecini), respectiv un sistem electroenergetic cu mai multe zone.

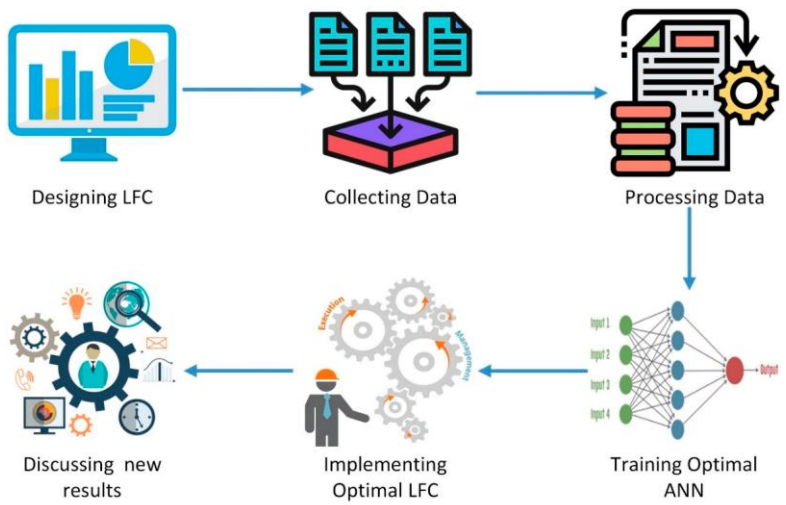



**Fig. 9.6.** Strategia de lucru a autorilor studiului din [52]

Diagrama de funcționare a unui regulator de frecvență a sarcinii este ilustrată în fig. 9.7.

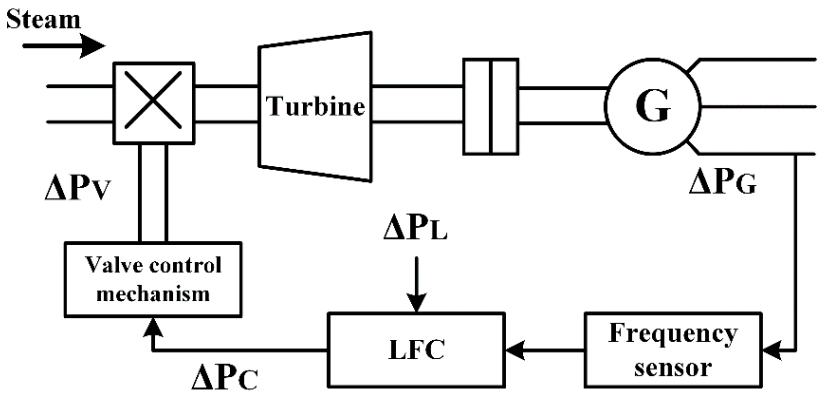



**Fig. 9.7.** Diagrama funcțională a regulatorului de frecvență a sarcinii [52]

Rețeaua neuronală folosită a fost optimizată prin utilizarea unui algoritm de optimizare, cu scopul de a reduce numărul de nueorni din stratul ascuns și a inițializa cu valori mai bune ponderile inițiale. Rezultatele au arătat că RNA optimizată a avut nevoie de un număr de 25 de epoci pentru antrenare, iar cea clasică 34 de epoci. Plus, numărul de neuroni sin stratul ascuns a scăzut de la 30 la 23. Rețeaua folosită este de tipul multistrat perceptron cu învățare supravegheată și învățare prin propagarea înapoi a erorii. Straturile RNA sunt stratul de intrare cu doi neuroni, stratul ascuns cu un număr variabil de neuroni, iar stratul de ieșire cu un singur neuron.

Rezultatele obținute prin simulare pentru metoda propusă au fost comparate cu rezultatele obținute cu o RNA simplă, respectiv cu un controler PID. Metoda propusă a dat un control mai stabil al frecvenței.

### 9.5.Aplicații în distribuția energiei electrice

Monitorizarea și creșterea nivelului calității energiei electrice este o problemă rezolvată prin implicarea rețelelor neuronale artificiale. O astfel de abordare este prezentată în articolul

[53] Mahar, H.; Munir, H.M.; Soomro, J.B.; Akhtar, F.; Hussain, R.; Elnaggar, M.F.; Kamel, S.; Guerrero, J.M. Implementation of ANN Controller Based UPQC Integrated with Microgrid. Mathematics 2022, 10, 1989. https://doi.org/10.3390/math10121989.

Studiul prezentat în articolul menționat propune utilizarea RNA pentru a crește calitatea energiei electrice prin integrarea unui UPQC la o microrețea conectată la unități de generare care folosesc resurse regenerabile. Astfel, se

propune un controler bazat pe RNA pentru a comonda UPQC-ul, astfel nu mai sunt necesare operații matematice complexe, iar costul controlerului scade mult. În plus, se utilizează sarcini neliniare dezechilibrate și tensiunea de alimentare armonică pentru a evalua performanța sistemului UPQC- generator fotovoltaic-baterie. De asemenea, sunt luate în considerare și alte probleme de calitate a energiei electrice precum variații ale tensiune, cum ar fi goluri de tensiune și supratensiuni. În studiu, sistem de control RNA este antrenat prin tehnica de retropropagare Levenberg-Marquardt pentru a furniza semnale de referință eficiente și a menține tensiunea necesară a condensatorului de curent continuu (a UPQC). Pentru testarea și validarea metodologie propuse, este utilizat MATLAB/Simulink, unde se efectuează simulări ale UPQC-ului fotovoltaic-baterie folosind control bazat pe SRF. Diagrama structurală a sistemului menționat anterior în [61] este prezentată în fig. 9.8.

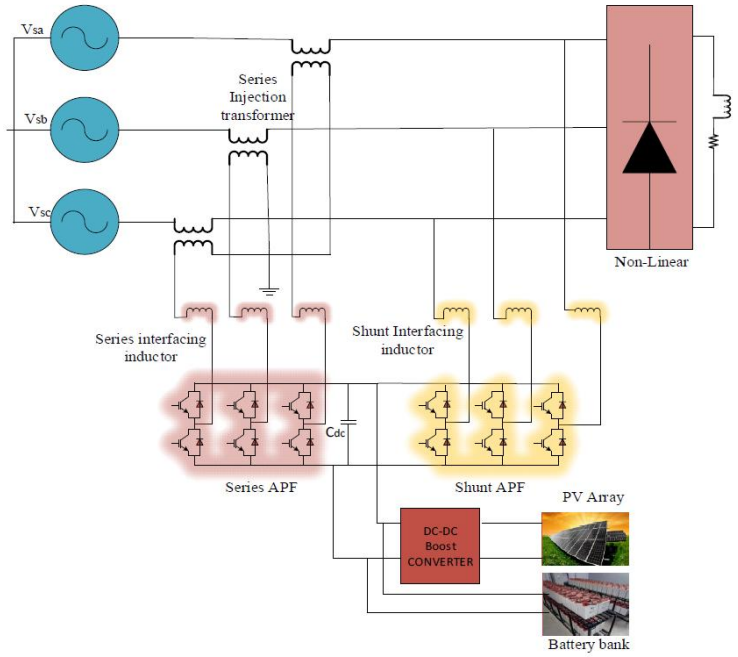



**Fig. 9.8.** Diagrama structurală a sistemului generator fotovoltaic-baterie-UPQC [53]

În articol sunt descrise două controlere RNA pentru comanda filtrului paralel și a celui serie din structura UPQC. În ambele situații sunt folosite RNA feedforward, cu două straturi ascunse, activarea neuronilor prin funcția sigmoid, iar deviația valorilor prezise față de cele reale este determinată prin utilizarea

indicatorului RMSE. Fig. 9.9 ilustrează controlul și arhitectura RNA a filtrului paralel, iar fig. 9.10 a filtrului serie.

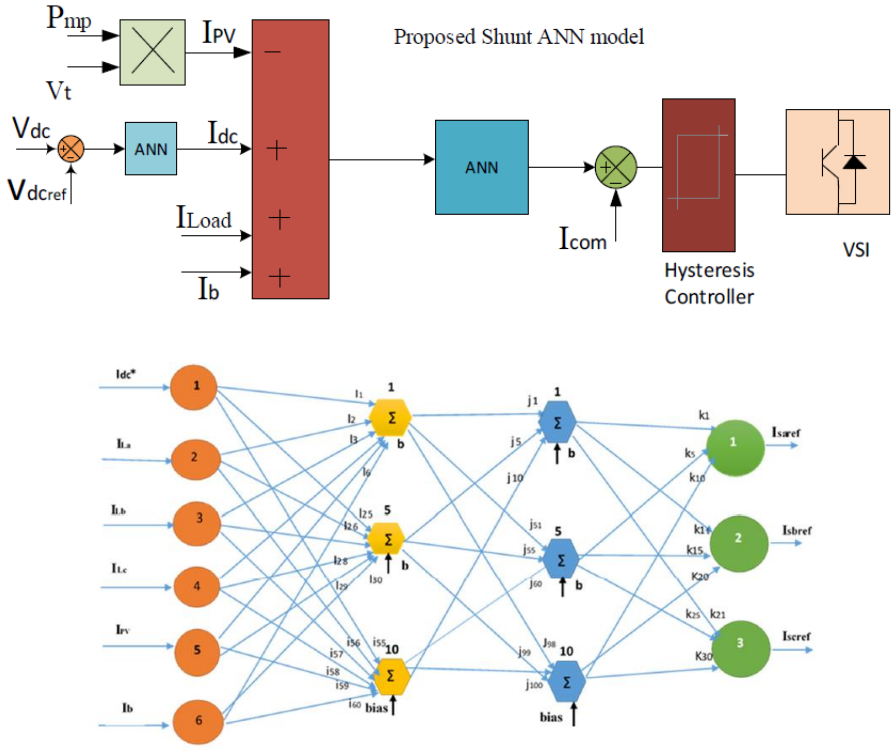



**Fig. 9.9.** Controlul și arhitectura RNA pentru filtrul paralel [53]

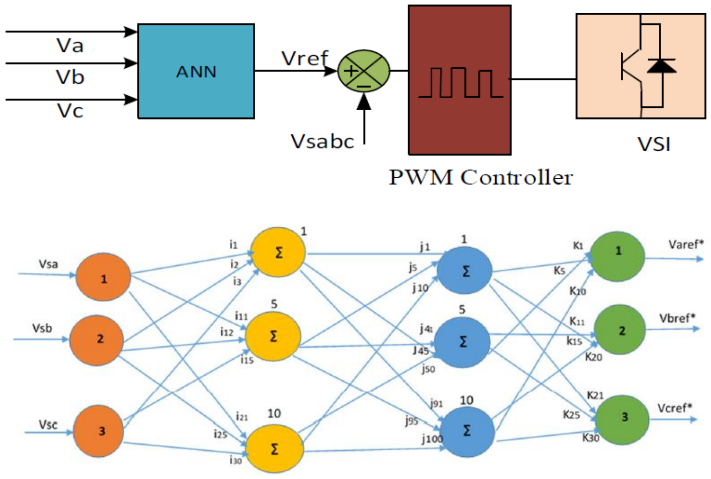



**Fig. 9.10.** Controlul și arhitectura RNA pentru filtrul serie [53]

Din fig. 9.9 se concluzionează următoarele: sunt folosite două RNA pentru a obține semnalele de comandă a unui controler histerezis, care va controla invertorul VSI. Variabilele de intrare ale primei RNA sunt tensiunile continue ale condensatorului, iar ieșirea este curentul continuu. Intrările pentru a doua RNA sunt curenții de fază, curentul continuu primit de la pasul anterior și curenții de la generatorul fotovoltaic, respectiv baterie, iar ieșirile sunt reprezentate de curenții de referință ai controlerului histerezis. Din arhitectura celei de a doua RNA se observă 6 neuroni de intrare, corespunzători celor 6 curenți menționați, câte 10 neuroni în fiecare dintre cele două straturi ascunse și trei neuroni la ieșire.

În fig. 9.10 controlul RNA este folosit pentru a comanda invertorul VSI serie cu ajutorul unui controler PWM. Rețeaua folosită are trei neuroni la intrare pentru cele trei tensiuni de linie, câte 10 neuroni ascunși în cele două straturi ascunse și trei neuroni la ieșire, care oferă valorile de referință ale tensiunilor contru comanda controlerului menționat.

În cele două figuri se observă că toți neuroni din straturile ascunse au asociate ponderi și valori bias.

Testarea cu ajutorul Simulink a metodologiei propuse în articol a indicat superioritatea acesteia față situația fără UPQC, când nivelul distorsiunilor armonice a fost menținut în limitele impuse prin standard.

Rețelele neuronale artificiale sunt folosite în managmenetul energiei pe componenta de distribuție a energiei electrice, din sistemele electroenergetice pentru a realiza sisteme de control. O astfel de aplicație a RNA este prezentată în articolul

[54] Irfan, M.M.; Malaji, S.; Patsa, C.; Rangarajan, S.S.; Hussain, S.M.S. Control of DSTATCOM Using ANN-BP Algorithm for the Grid Connected Wind Energy System. Energies 2022, 15, 6988. https://doi.org/ 10.3390/en15196988.

Unde autorii propun utilizarea unei rețele de tipul multistrat perceptron cu propagarea înapoi a erorii pentru a controla un DSTATCOM în rețeaua de transport și astfel a menține calitatea energiei electrice în intervalul limită cerut în standarde.

Structura RNA propuse în lucrare este prezentată în fig. 9.11. Se observă în imagine că rețeaua are trei straturi de neuroni (intrare, ascuns și ieșire), cu câte trei neuroni pe fiecare strat. Rețeaua este complet conectată. La intrare, rețeaua primește curenții pe cele trei faze, iar la ieșire va oferi curenții cu care se poate

controla un VSI (Voltage Source Inverter – Invertor sursă de tensiune), mai exact impulsurile de comutare ale acestuia.

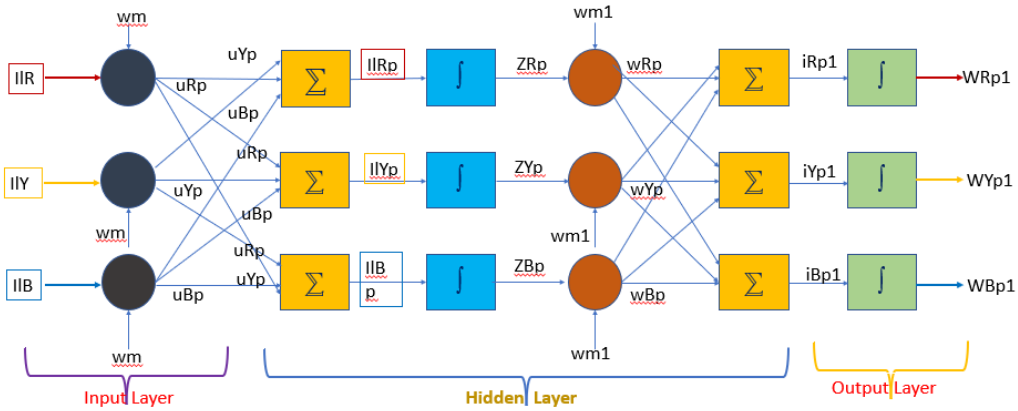



**Fig. 9.11.** Structura RNA propusă în [54]

În lucrare sunt explicate toate etapele care trebuie realizate de către RNA, plus baza matematică.

În concluzie, un controler RNA cu backpropagation este modelat pentru a obține impulsurile de poartă ale invertorului conectat la rețeaua de transport. VSI conectat în paralel este considerat un DSTATCOM și astfel direcționează puterea reactivă necesară către sarcina asociată rețelei și apoi reduce armonicele produse la nodurile de conectare.


**Fig. 9.12.** Stand experimental [54]

Acest controler RNA a fost implementat pe un FPGA, care la rândul lui a fost integrat într-un stand experimental prezentat în fig. 9.12.

La finalul studiului, rezultatele obținute prin simulare au fost comparate cu cele obținute prin standul experimental; iar rezultatele simulate au fost mai bune decât cele obținute din sistemul fizic.

### 9.6.Aplicații în utilizarea energiei electrice

Utilizarea calculului neuronal în sistemele electroenergetice din sectorul consumatorilor de energie electrică se face în problemele care implică cunoașterea din avans a consumului de energie electrică, adică prognoză. O astfel de aplicație este prezentată în articolul

[55] Waseem, M.; Lin, Z.; Yang, L. Data-Driven Load Forecasting of Air Conditioners for Demand Response Using Levenberg–Marquardt Algorithm-Based ANN. Big Data Cogn. Comput. 2019, 3, 36. https://doi.org/10.3390/bdcc3030036.

Autorii acestui articol propun utilizarea unui multistrat perceptron pentru a realiza prognoza consumului de energie a unor aparate de aer condiționat folosindu-se de o baza mare de date colectate. Față de exemplele anterioare, autorii folosesc algoritmul LMA pentru a actualiza ponderile rețelei. Acest aspect se observă în fig. 9.13.

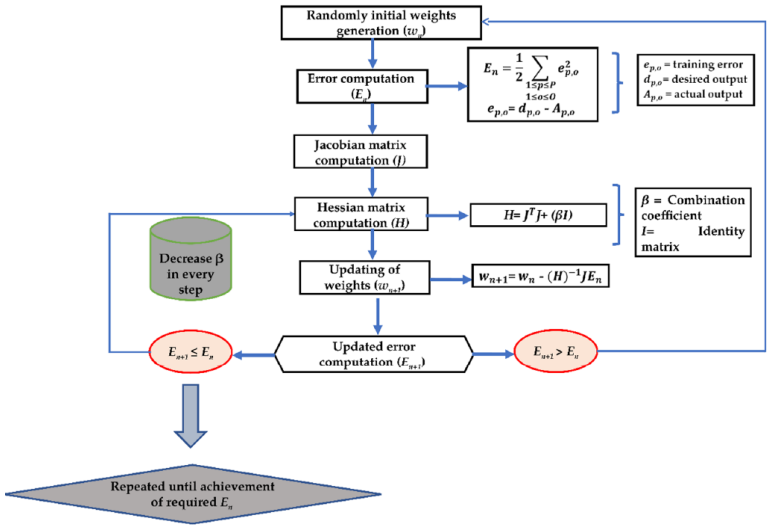



**Fig. 9.13.** Algoritmul LMA folosit pentru antrenarea RNA propuse [55]

Pentru analiza performanței rețelei și a rezultatelor prezise se folosesc trei indicatori, MSE, MAPE și MAE (fig. 9.14). Datele de intrare ale RNA propuse sunt timpul, datele meteo și informații despre consumul aparatelor. Diagrama

funcțională din figura de mai jos ne arată faptul că datele preluate sunt procesate, înainte de a fi transmise rețelei. Din aceste date, autorii folosesc 75% pentru antrenare, și 25% pentru validare și testare.

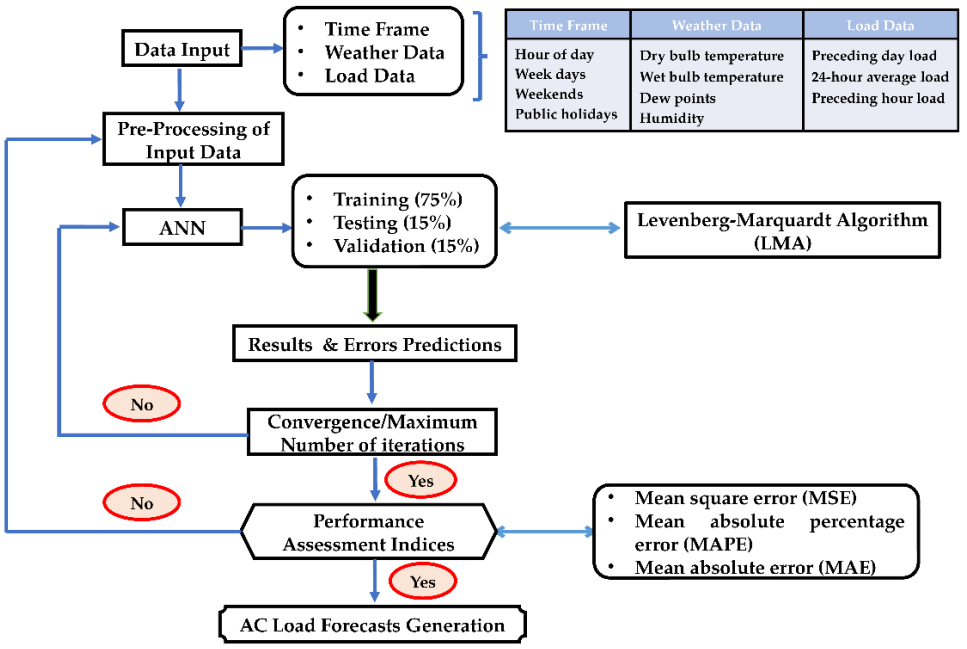



**Fig. 9.14.** Diagrama funcțională a RNA propuse [55]

Implementarea metodei propuse s-a realizat prin intermediul MATLAB, iar rezultatele obținute subliniază superioritatea metodei propuse comparativ cu alte metode.

Un alt exemplu de utilizare a RNA în managementul energiei electrice în sectorul consumatorilor este studiul prezentat în articolul

[56] Kim, S.; Jung, D.-Y. Fault Estimation of Rack-Driving Motor in Electrical Power Steering System Using an Artificial Neural Network Observer. Electronics 2022, 11, 4149. https://doi.org/10.3390/electronics11244149.

În această lucrare se propune estimarea defecțiunii unui motor într-un sistem de servodirecție electrică de tip cremalieră utilizând un observator (RNA). Diagrama funcțională a sistemului de estimare este descrisă în fig. 9.15.

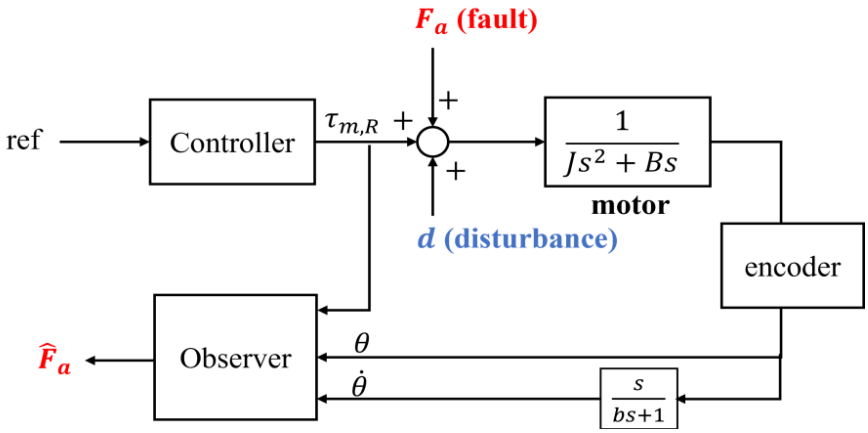



**Fig. 9.15.** Diagrama funcțională a sistemului pentru estimarea defecțiunilor [56]

Arhitectura rețelei neuronale este ilustrată în fig. 9.16, cu un strat de intrare, trei straturi ascunse și un strat de ieșire. Datele de intrare sunt poziția unghiulară a motorului, viteza unghiulară și cuplul de control, precum și eroarea absolută dintre traiectoria dorită și poziția unghiulară reală a motorului. Aceste date de intrare sunt normalizate pentru a fi corespunzătoare cerințelor rețelei. La ieșire este un singur neuron. Straturile ascunse au un număr diferit de neuroni, adică primele două au același număr, iar al treilea este diferit. Aceste numere sunt variate în timpul studiului pentru a determina structura ideală pentru aplicația dedicată. Neuronii asociați straturilor ascunse au ponderile și bias-urile corespunzătoare.

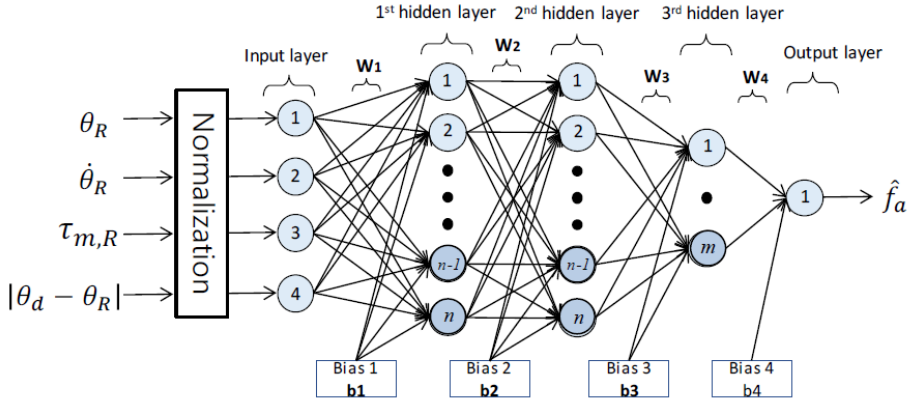



**Fig. 9.16.** Arhitectura RNA propuse pe post de observator [56]

Implementarea metodologiei propuse s-a realizat prin implicarea MATLAB, și implementarea fizică, precum arată fig. 9.17.

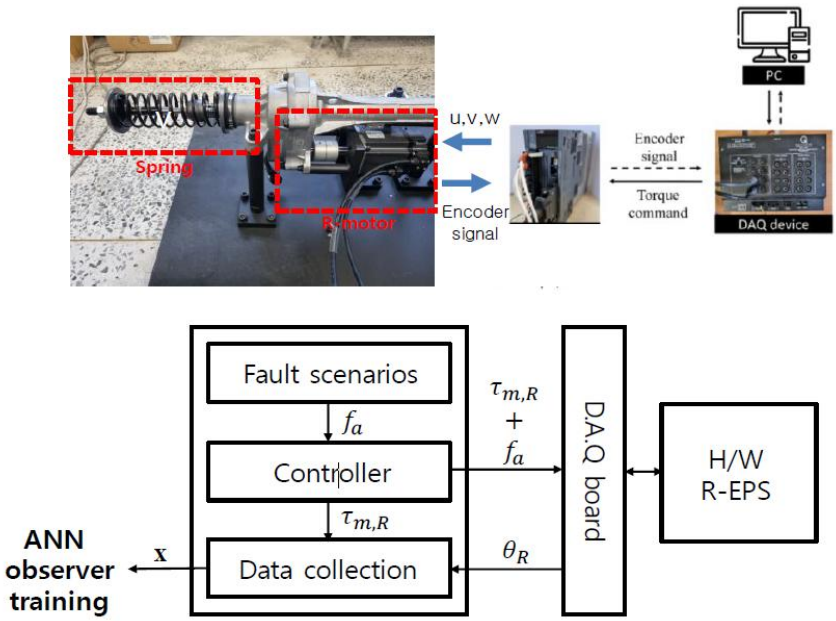



**Fig. 9.17.** Implementarea experimentală a metodei propuse [56]

În imaginea din fig. 9.17 se observă etapa de colectare a datelor, precum și motorul și controlul motorului.

În articol se validează metoda propusă prin comprarea acesteia cu alte metode din literatura de specialitate.

În sistemele electroenergeticee, RNA sunt folosite și în sistemele de control. Un astfel de exemplu este articolul

[57] Kim, T.-G.; Lee, H.; An, C.-G.; Yi, J.; Won, C.-Y. Hybrid AC/DC Microgrid Energy Management Strategy Based on Two-Step ANN. Energies 2023, 16, 1787. https://doi.org/10.3390/en16041787.

În care două RNA în cascadă sunt elementele componente ale unui sistem de management al energiei (Energy Management System) dedicat controlului circulației de putere într-o microrețea hibridă CC/CA, care conține generatoare eoliene, fotovoltaice și unități de stocare (fig. 9.18), pe partea de CC, și Aceleași componente plus centrală de energie electrică pe partea de CA. Cele două componente ale microrețelei sunt conectate între ele printr-un convertor.

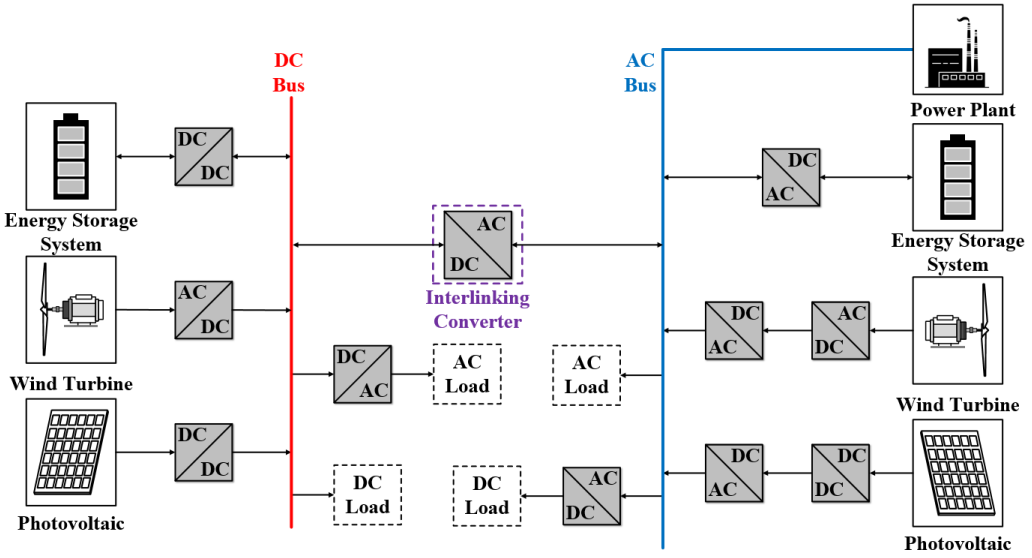



**Fig. 9.18.** Structura unei microrețele hibride CC/CA [57]

Procesul de control folosit în studiu este unul centralizat, a cărui funcționare este prezentată în fig. 9.19. Se observă în imagine că fiecare element al microrețelei este controlat printr-un regulator local, iar acestea sunt controlate de un controler central. Avantajul acestei abordări este că controlul puterii se realizează foarte ușor prin unitățile de stocare.

Metodologia propusă în articol, a fost implementată fizic, la o scară redusă, experimentată în laborator. În lucrare se explică în detaliu componența sistemului experimental, și structura controlerului central, care cuprinde cele două RNA în cascadă (fig. 9.20). În figură se observă că cele două rețele sunt de tipul multistrat perceptron cu un singur strat ascuns. Prima rețea dictează modul de operare a microrețelei, iar a doua comanda puterii microrețelei. Ambele rețele au câte trei neuroni de intrare, 20 neuroni ascunși și un neuron de ieșire.

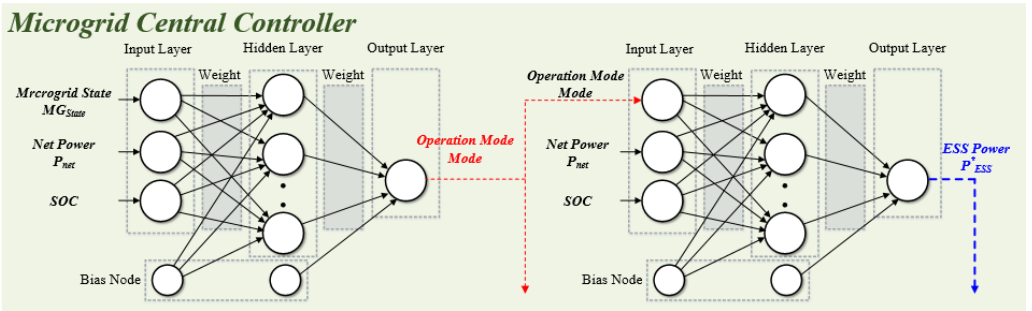



**Fig. 9.20.** Structura controlerului central bazat pe RNA [57]

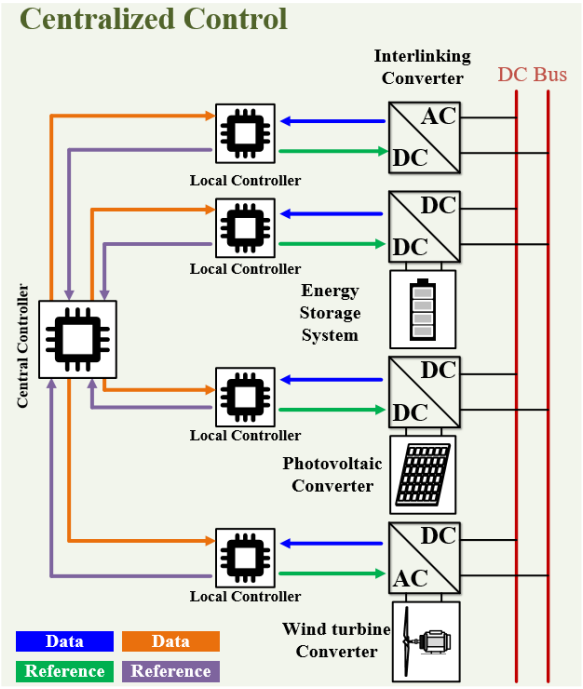



**Fig. 9.19.** Controlul centralizat a unei microrețea hibridă [57]

În articol se explică fiecare etapă a procesului de control, și implementarea experimentală de laborator. Concluziile studiului sunt că utilizarea RNA pentru a controla modul de funcționare a unei microrețele este fezabil. Mai mult, în studiul experimental, s-a confirmat că modul de operare, care este rezultatul primei etape/RNA, și comanda de putere a unității de stocare a energiei, care este rezultatul celei de-a doua etape/RNA, funcționează stabil în condiții de sarcini variabile și diferite puteri introduse de unitățile de generare distribuite, iar convertoarele individuale sunt controlate de convertorul central.

Un alt tip de problemă din managementul energiei, în a cărui rezolvare a fost implicat calculul neuronal este eficiența în procesul de încărcare a vehiculelor electrice. În acest sens, articolul

[58] Abrantes, G.C.; Costa, V.S.; Perdigão, M.S.; Cruz, S. Mutual Inductance Estimation Using an ANN for Inductive Power Transfer in EV Charging Applications. Energies 2024, 17, 1615. https://doi.org/10.3390/en17071615

prezintă un studiu a cărui obiectiv principal a fost estimarea inductanței mutuale folosind RNA pentru transferul inductiv de putere în aplicațiile de încărcare a vehiculelor electrice.

Un sistem de încărcare a vehiculelor electrice prin transfer inductiv de putere este ilustrat în fig. 9.21. Din această imagine se poate concluziona că poziția vehiculului va influența valoarea inductanței mutuale, și modul de transmitere a energiei, deci eficiența tranferului de putere.

Studiul a fost realizat prin implementarea metodologie în Simulink, și testarea eficienței abordării propuse prin simuări. Schemele electrice luate în considerare pentru modelarea sistemului de încărcare și determinarea inductanței mutuale sunt descrise în fig. 9.22.

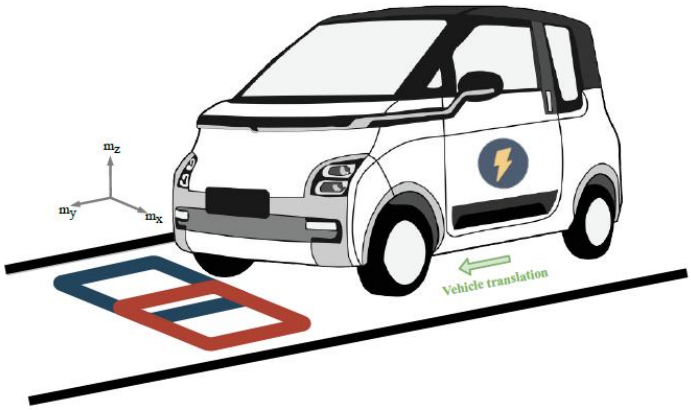



**Fig. 9.21.** Alimentarea prin transfer inductiv de putere a vehiculelor electrice [58]

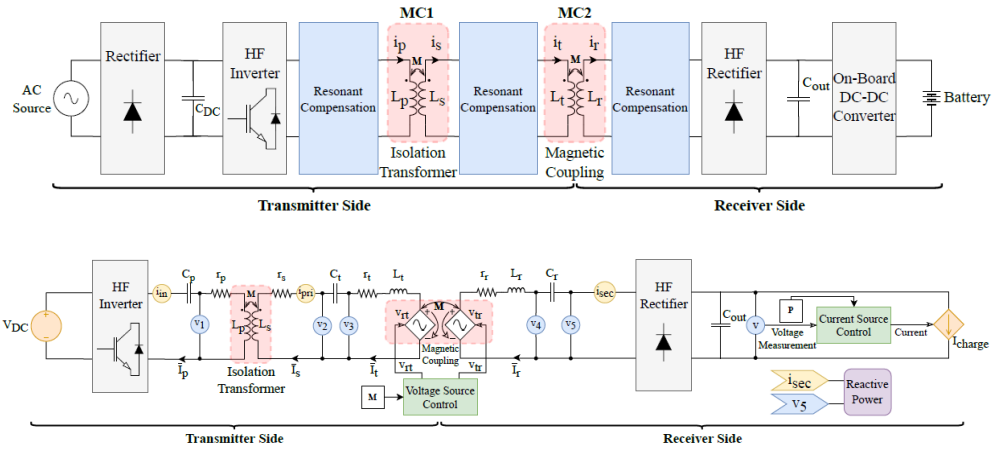



**Fig. 9.22.** Schema de principiu a sistemului simulat - inductanța mutuală [58]

Arhitectura RNA folostă pentru estimarea inductanței este cea din fig. 9.23. Rețeaua utilizată este de tipul mutistrat perceptron, cu două straturi ascunse. Stratul de intrare are 7 neuroni, care primesc informații despre valoarea tensiunilor și a curenților armonici, curenți care influențează inductanța mutuală.

Straturile ascunse sunt diferite, în sensul că primul strat ascuns are 1024 de neuroni, iar al doilea unul. Stratul de ieșire are un singur neuron. Toți neuronii rețelei sunt complet conectați cu neuronii din straturile superioare.

Antrenarea RNA a fost realizată folosind date obținute din simulări, iar pentru evaluarea deviației, indicatorul MSE (Mean Square Error). Antrenarea și validarea a fost realizată pe 200 de epoci.

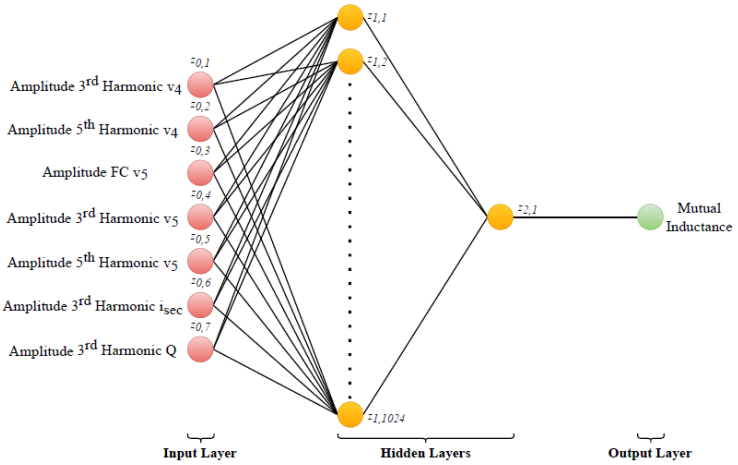



**Fig. 9.23.** Arhitectura RNA dedicată estimării inductanței mutuale [58]

La final, rezultatele obținute prin metoda propusă au fost comparate cu rezultatele obținute în literatura de specialitate, comprația indicând superioritatea metodei din articol.

## 9.7. Întrebări și exerciții

1. Aplicații ale RNA în managementul energiei se focalizează pe creșterea eficienței energetice.
   Adevărat / Fals

2. Implementarea RNA cea mai populară este
   RNA software / RNA hardware / RNA hibrid

3. Utilizarea RNA în producerea energiei electrice se realizează sub formă sisteme de prognoză și control
   Adevărat / Fals

4. Exemplele date pentru utilizarea RNA în producerea energiei electrice sunt în sisteme de control și estimare
   Adevărat / Fals

5. Utilizarea RNA în distribuția energiei electrice și exemplificate în capitol sunt în constrol și prognoză
   Adevărat / Fals

6. Utilizarea RNA în sectorul de consum al energiei electrice este pentru obținerea de prognoze și estimări.
   Adevărat / Fals

# 10

# Algoritmi genetici

Acest capitol introduce subiectul principal al următoarelor trei capitole, și anume o nouă tehnică de inteligență artificială, Algoritmii Genetici, AG. Astfel, în prima parte a capitolului se prezintă aspectele generale ale algoritmilor genetici, urmată de prezentarea istoriei și a atributelor de bază ale AG.

## 10.1. Introducere

Algoritmii genetici fac parte din marea familie a tehnicilor de căutare/optimizare, ele intrând în domeniul IA prin faptul că au la bază teoria evoluționistă darwiană.

Prin implicarea tehnicilor de căutare / optimizare se urmărește îmbunătăţirea performanţelor unui sistem, către un punct optim" (care asigură performaţe maxime). Din această afirmație putem extrage două componente distincte ale unei tehnici de optimizare, și anume (1) procesul de căutare (funcția de optimizare) și (2) punctul de optim. Dar, în cele mai multe cazuri tehnicile de căutare se focalizează pe atingerea punctului optim şi mai puţin pe calea folosită pentru atingerea acestuia.

Tehnicile de căutare se clasifică în trei mari categorii:

- tehnici de calcul numeric,
- tehnici enumerative (exhaustive),
- tehnici de căutare aleatorie.

Fig. 10.1. prezintă succint cele trei clase de tehnici de căutare și subclasele acestora.

*Tehnicile de calcul numeric* se împart la rândul lor, în metode directe şi metode indirecte. Metodele directe apelează la o serie de ipoteze simplificatoare

– cel mai adesea, ipoteza unei forme pătratice a suprafeţei funcţiei de optimizat, care permit calculul direct, imediat al punctului de optim. Aceste metode sunt eficiente numai în măsura în care ipotezele adoptate sunt suficient de precise. Metodele indirecte, de tipul metodelor de gradient, urmăresc apropierea de optimele locale urmând căi de gradient maxim. Ambele metode menţionate fac parte din categoria celor mai studiate tehnici de optimizare, dar ele au robusteţe slabă, întrucât în multe cazuri practice suprafaţa funcţiei de optimizare se caracterizează prin numeroase discontinuităţi şi prezenţa unor multiple puncte de optim local.

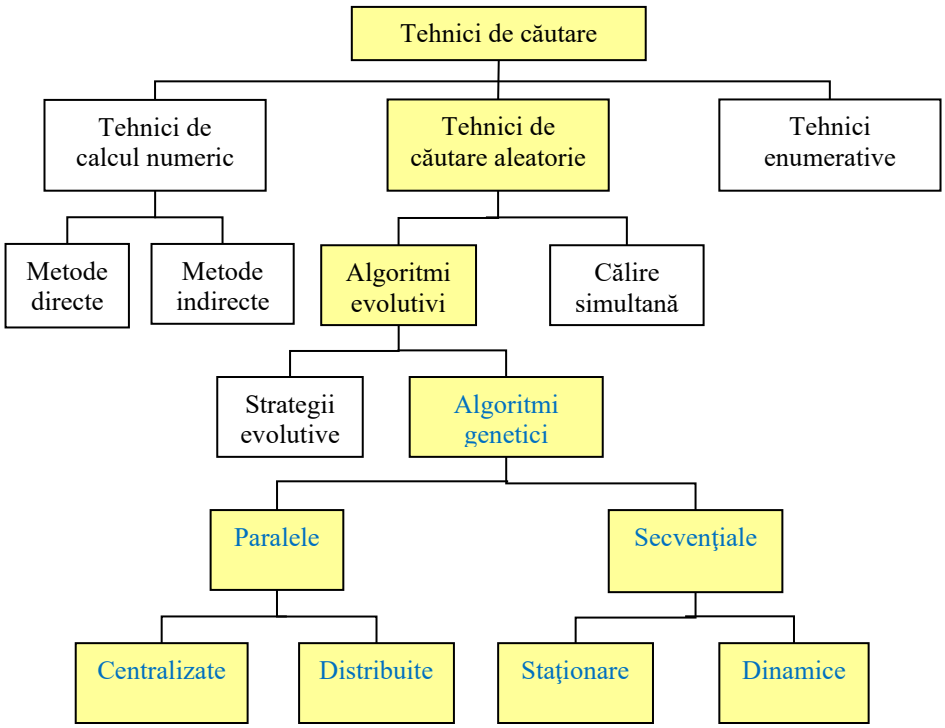



**Fig. 10.1.** Tehnici de căutare

*Tehnicile enumerative / exhaustive* pot fi folosite în cazul unor spaţii de căutare finite sau infinite, dar discretizate. Asemenea tehnici consideră fiecare punct din spaţiu de căutare şi evaluează funcţia de optimizat în acel punct, selectând în cele din urmă punctul de valoare maximă/minimă a funcţiei respective. Aceste tehnici se dovedesc eficiente numai în cazul unor probleme simple, pentru care spaţiul de căutare este suficient de „îngust”.

*Tehnicile de căutare aleatorie* au captat atenţia ca reprezentând posibile răspunsuri pentru inconvenientele tehnicilor de calcul numeric şi enumerative.

Dar, căutarea aleatorie simplă este la fel de ineficientă ca tehnicile enumerative. Pentru creşterea eficienţei tehnicilor de optimizare aleatorie, este necesar ca la componenta aleatorie să se suprapună o abordare euristică, care să orienteze procesul de căutare către direcţii „promiţătoare".

Una dintre aceste tehnici este *călirea simultană*, care permite deplasarea către soluţii mai bune sau mai slabe cu o probabilitate proporţională cu caracterul soluţiei. Această posibilitate de deplasare, la un moment dat, către soluţii mai slabe permite „eliberarea" din optimele locale. În acelaşi timp, probabilitatea deplasării către o soluţie mai slabă scade în timp, astfel încât, iniţial căutarea are un caracter global, iar în partea finală capătă un caracter local.

Tot în categoria tehnicilor de căutare aleatorie intră şi algoritmii evolutivi, care sunt precursorii algoritmilor genetici. Aceştia din urmă folosesc selectarea aleatorie a direcţiei de deplasare pe baza unor operatori specifici (selecţie, încrucişare şi mutaţie).

Algoritmii genetici sunt tehnici robuste de căutare, de cele mai multe ori mai eficiente decât majoritatea tehnicilor de căutare, datorită caracteristicilor lor:

- Nu folosesc variabilele funcţiei de optimizare, ci codificări ale acestora;
- Procesul de căutare capătă caracter paralel, explorându-se simultan mai multe domenii din spaţiul de căutare;
- Deplasarea dintr-un punct în altul se face pe baza unor reguli probabilistice şi nu deterministe;
- Determinarea direcţiilor de deplasare se face doar pe baza evaluării funcţiei de adaptare, fără a fi necesară cunoaşterea derivatelor acesteia.

Pentru ca procesul de căutare să fie cât mai eficient, un AG trebuie să se desfăşoare respectând două etape: explorarea şi exploatarea.

Etapa de explorare se caracterizează printr-o căutare foarte puţin direcţionată, astfel încât să fie posibilă cercetarea unui număr cât mai mare de domenii din spaţiul problemei, în căutarea domeniului mai „promiţător", în care să se concretizeze apoi căutarea mai aprofundată.

În cursul etapei de exploatare, domeniul izolat rezultat în etapa anterioară urmează a fi cercetat pentru determinarea soluţiei optime. În acest context, forma standard a AG acţionează foarte puţin timp în zona de explorare, consumând cea mai mare parte a timpului de calcul pentru etapa de exploatare. Acest lucru se manifestă cu atât mai puternic cu cât operatorul de încrucişare este folosit mai intensiv, deoarece încrucişarea determină reducerea diversităţii populaţiei. Pentru o comportare cât mai bună a AG se impune luarea unor măsuri care să

permită prelungirea etapei de explorare, măsuri care – în practică – se traduc cel mai adesea prin creşterea dimensiunii populaţiei şi inhibarea încrucişărilor în faza iniţială, diversificarea asigurată doar de operatorul mutaţie.

## 10.2. Istoria algoritmilor genetici

Algoritmi genetici au fost creaţi şi dezvoltaţi de către John Holland, de la Universitatea din Michigan, în anii ’70 şi ulterior de către David Goldberg şi K.A.De Jong.

Dezvoltarea noilor paradigme a fost o consecinţă a cercetărilor făcute pentru îndeplinirea a două obiective:

- abstractizarea şi explicarea cât mai riguroase a proceselor adaptive ale sistemelor naturale.
- crearea unor sisteme artificiale care să păstreze robusteţea sistemelor naturale.

Începuturile algoritmilor genetici sunt dinainte de 1950, ei fiind inspiraţi din teoria evoluţionistă a lui Charles Darwin. Conform acestei teorii, toate sistemele biologice se dezvoltă prin mecanismele evoluţiei şi selecţiei naturale. Astfel, acest fapt a dus la crearea unor structuri biologice foarte complexe care, din punct de vedere al naturii, au un singur obiectiv - auto – reproducerea. În contextul selecţiei naturale, sistemele care se pot reproduce cu succes vor prolifera pe seama celor care nu se pot reproduce sau o fac mai puţin performant. Conform teoriei evoluţioniste, pentru ca aceasta să aibă loc, orice sistem trebuie să satisfacă trei condiţii: (1) să fie apt de reproducere, (2) structurile „slabe” să poată fi eliminate, (3) reproducerea să fie însoţită de anumite „erori” (mutaţiile) care să asigure diversificarea structurilor.

Următoarea etapă a evoluției AG a fost între anii 1950 și 1960. În anii 1950, Alan Turing a propus un proces de căutare genetică în contextul inteligenței artificiale. În aceeași perioadă, biologii și informaticienii au început să exploreze simularea evoluției pe computere, cu toate acestea, modelele computaționale formale erau încă subdezvoltate, fapt care s-a întâmplat după această perioadă.

Între anii 1960-1975, John Holland, profesor la Universitatea din Michigan, care este considerat părintele algoritmilor genetici, a lucrat la un studiu, care în final s-a concretizat prin publicarea cărții de referință a algoritmilor genetici ”Adaptarea în sistemele naturale și artificiale”. În această carte s-a formalizat modelul algoritmilor genetici, și s-au introdus operatorii genetici selecția,

încrucișarea și mutația ca operatori principali, precum și termeni specifici precum conceptul de schemă și teorema schemei.

Dezvoltarea AG a fost continuată de studenții lui J. Holland între anii 1970 și 1980. Astfel, David E. Goldberg a folosit AG pentru optimizare în sistemele de control și inginerie. Cartea sa din 1989, „Algoritmi genetici în căutare, optimizare și învățare automată", a devenit un material clasic în studiul algoritmilor genetici. Alți cercetători, precum Kenneth De Jong, au contribuit la analiza performanței AG. Teza de doctorat a lui De Jong din 1975, a fost primul studiu empiric cuprinzător privind performanța acestor algoritmi.

Anii 1980-1990 a reprezentat perioada de expansiune și popularitate a acestor tehnici de IA. Într-adevăr, algoritmii genetici au câștigat popularitate în proiectarea inginerească, programare și planificare, robotică și optimizarea funcțiilor. În plus față de perioada anterioară, domeniul calculului evolutiv s-a extins pentru a include programarea genetică (Koza, anii 1990), strategiile evolutive (Rechenberg și Schwefel), programarea evolutivă (Fogel). În final, algoritmii genetici au devenit cunoscuți pentru rezolvarea problemelor dificile, neliniare, multimodale, de tip „cutie neagră".

După anii 2000 a avut loc dezvoltarea algoritmilor genetici prin integrarea lor cu alte tehnici de IA precum logică fuzzy, rețele neuronale, și alte tehnici de optimizare, rezultând metode meta-euristice hibride. Mai mult, domeniile de aplicare ale AG s-au înmulțit, prin utilizare în bioinformatică, modelare financiară, reglarea hiperparametrilor în învățarea automată și optimizarea energiei regenerabile. În această problemă, framework-uri moderne precum DEAP, GAUL și integrarea în bibliotecile MATLAB, Python și Java au menținut algoritmii genetici printre tehnicile de IA utilizabile pe scară largă.

O sinteză a istoriei AG este ilustrată în fig. 10.2.

În concluzie, algoritmii genetici încearcă să imite natura (selecția naturală), dar în loc să construiască structuri – soluţii, le dezvoltă pornind de la un set de structuri / soluţii iniţiale.

Între cele două concepte, selecția naturală evoluționistă (teoria evoluționistă) și algoritmi genetici există diferenţe fundamentale. În ceea ce privește timpul și numărul de indivizi din populații, evoluţia naturală s-a produs de-a lungul unei populaţii extinse, pe o perioadă foarte lungă de timp, iar algoritmii genetici folosesc populaţii de dimensiuni reduse (sute sau mii de cromozomi). Astfel, AG pierd o calitate remarcabilă a sistemelor naturale - ceea ce pare puţin probabil pe termen scurt, se poate transforma în certitudine pe termen lung. Dacă se ia în considerare modul de obținere a noilor membrii, selecţia naturală este un proces

competitiv, deoarece membrii unei populaţii se află permanent în competiţie pentru resurse şi pot adopta strategii diverse pentru o cât mai bună adaptare.

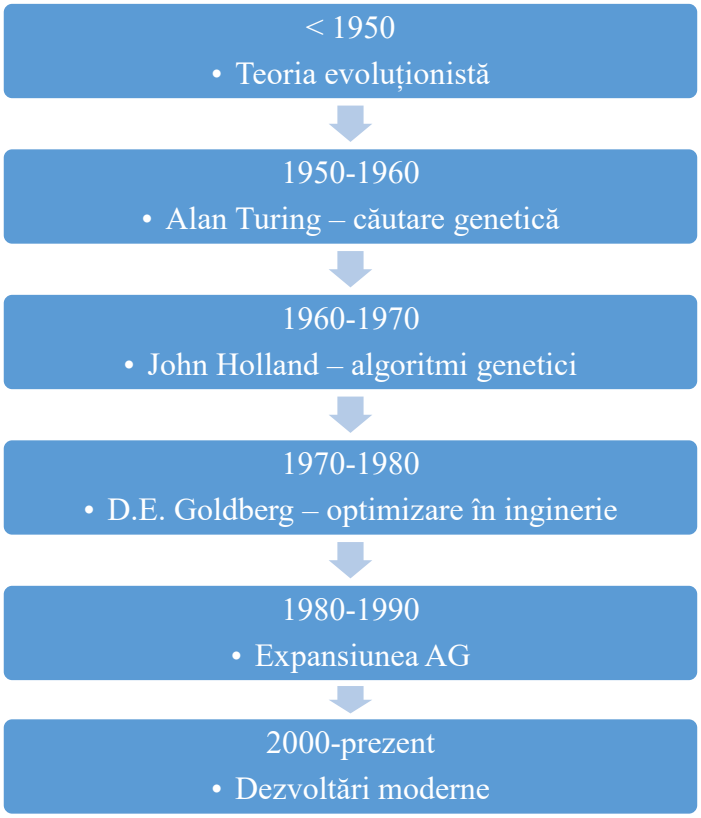



**Fig. 10.2.** Istoria algoritmilor genetici

Selecţia naturală se produce într-un mediu aflat într-o continuă schimbare, iar dimpotrivă, în cazul AG, obiectivul urmărit este de obicei identificarea unei singure soluţii într-un mediu şi într-un interval de timp prescrise şi mai puţin promovarea competiţiei pentru un progres continuu.

În natură, selecția naturală folosește încrucişarea diploidă – fiecare urmaş conţine două seturi de cromozomi, câte unul de la fiecare părinte, iar o genă se poate manifesta prin dominanţă (una din cele două gene o domină pe cealaltă şi se manifestă singură) sau prin dominaţă incompletă (cele două gene sunt egale, iar urmaşul moşteneşte, pentru aceeaşi caracteristică, trăsături parţiale de la ambi părinţi). În cazul AG, moştenirea este de tip haploid, adică urmaşul conţine un singur cromozom, iar genele acestuia provin nealterate de la ambii părinţi, cu alte cuvinte, genele se manifestă întotdeauna prin dominanţă.

### 10.3. Atributele algoritmilor genetici

Algoritmii genetici (așa cum s-a prezentat în subcapitolele anterioare) sunt algoritmi de căutare care au la bază mecanismele specifice geneticii şi selecţiei naturale. În consecință, indivizii cei mai bine adaptaţi dintr-o populaţie manifestă tendinţa de a se reproduce şi a supravieţui în generaţia următoare, asigurând ân acelaşi timp perpetuarea calităţilor lor şi îmbunătăţirea globală a calităţii generaţiilor următoare. Contrar, indivizii mai slab adaptaţi îşi pot schimba structura şi, prin reproducere, trec şi ei în generaţia următoare. Rezultatele obţinute până în prezent au demonstrat că algoritmii genetici permit rezolvarea problemelor de natură liniară sau neliniară prin explorarea întregului spaţiu de căutare şi exploatarea zonelor promiţătoare, folosind cei trei operatori de bază: selecţia, încrucişarea şi mutaţia.

Termenii folosiţi în algoritmii genetici provin din biologie şi genetică:

- AG lucrează cu o ***populaţie*** de cromozomi, fiecare dintre aceştia reprezentând o soluţie posibilă a problemei de optimizare analizate. Deci, populația reprezintă mulțimea de soluții posibile, cu care algoritmul genetic lucrează. Astfel, aceste tehnici de IA lucrează cu soluțiile pentru a determina cea mai bună soluție, contrar celorlalte tehnici de IA, care funcționează pentru ca la final să obțină soluția optimă.
- ***Cromozomul*** este format din elemente denumit ***gene***, care exprimă o anumită caracteristică a fenotipului (codificarea soluției), fig. 10.3.
- Informaţia înmagazinată în genă poate lua anumite valori, denumite ***alele.*** Valorile alelelor pot fi binare, naturale, reale, sau simboluri (litere).
- Obținerea de noi cromozomi se face prin implicarea unor operatori similari cu evenimentele naturale, adică ***selecția***, ***reproducerea*** și ***mutația***.
- Obținerea unei noi populații poartă numele de ***generație***.

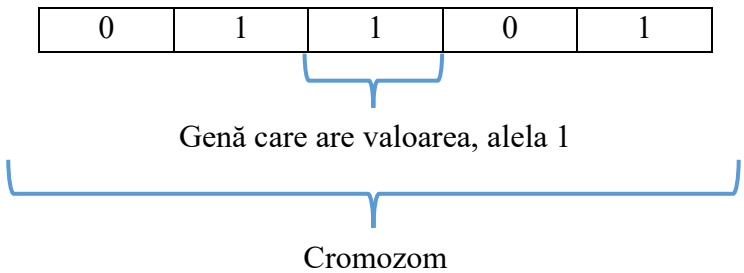



**Fig. 10.3.** Structura unui cromozom (codificare binară)

Uneori cromozomul este denumit ***genotip*** (codificarea soluţiei), care se deosebește de fenotip (soluţia însuşi). Genele se spune că sunt adaptate (la medii în care există populaţia respectivă). Cromozomii reprezintă lanţuri de simboluri sau numere (lanţuri binare, formate din elemente de 0 şi 1).

Adaptarea cromozomilor se determină prin folosirea unei ***funcții de adaptare*** (fitness function), iar apropierea unui cromozom față de soluția optimă se cuantifică prin funcția/***funcțiile obiectiv***.

Operatorii genetici sunt probabilistici, deci operațiile AG sunt stocastice, nu deterministe. Aceasta deoarece, selecția, încrucișarea și mutația utilizează alegeri aleatorii ghidate de probabilități.

Un alt atribut al AG este faptul că nu este necesar determinarea unui gradient, ceea ce face posibil folosirea unor funcții nediferențiabile, discrete sau afectate de zgomot. Deci, AG sunt utili pentru problemele de optimizare de tip „cutie neagră".

Un ultim atribut este paralelismul și căutarea globală, ceea ce implică faptul că AG susțin în mod natural execuția paralelă. În concluzie, algoritmii genetici tind să efectueze căutare globală, reducând șansa de a rămâne prinși în optimi locali.

### 10.4. Tipuri de algoritmi genetici

Există o mare varietate de algoritmi genetici, care se împart în opt categorii principale în funcție de modul în care sunt gestionate populația, selecția și reproducerea, respectiv cum este codificată soluția. În continuare sunt descrise succint cele opt categorii de AG:

1. Algoritmi genetici simpli, AGS – dezvoltați de J. Holland, se caracterizează prin codificarea binară, selecția prin metoda ruletei, reproducerea într-un punct sau două puncte, mutația prin schimbarea alelei la valoarea opusă, iar la fiecare generație, întreaga populație de cromozomi este schimbată.
2. Algoritmi genetici în stare staționară, AGSS – principala lor caracteristică este schimbarea doar a câțiva cromozomilor de la o generației la alta. Comparativ cu AGS, la acești algoritmi apare o convergență mai rapidă, adesea se folosește selecția prin turneu, iar în unele cazuri ajută la menținerea diversității soluțiilor.
3. Algoritmi genetici elitiști, AGE – la acești AG este adăugat elitismul pentru păstrarea celor mai adaptați dintre cromozomi de la o generație la

alta. Un avantaj important a acestor AG este faptul că nu suferă din cauza variațiilor aleatorii, care pot duce la pierderea celor mai bune soluții. Deasemenea, elitismul poate fi determinist (când un număr specifi de indivizi sunt selectați) sau probabilistic.

4. Algoritmi genetici paralel, AGP – populația este divizată și procesată în paralel pentru a accelera calculul și a menține diversitatea. Principalele trei tipuri de AGP sunt
   4.1. Master-Slave – evaluarea paralelă a adaptării,
   4.2. Modelul insuleleor – subpopulațiile (insulele) evoluează independent și ocazional scimbă indivizii (migrație),
   4.3. Modelul celular - Indivizii trăiesc pe o grilă și se împerechează cu vecinii (bun pentru menținerea diversității).
5. Algoritmi genetici adaptivi, AGA – caracteristic acestor AG este faptul că unii parametri, precum rata de mutație și rata de încrucișare sunt adaptați dinamic în timpul evoluției. Alte proprietăți imprtante ale acestor algoritmi sunt creșterea explorării atunci când este necesar și evitarea convergenței prematura.
6. Algoritmi genetici hibrizi, AGH – combină algoritmi genetici cu alte metode de optimizare (precum, căutarea locală, sau coborârea în gradient). Mai mult, acești algoritmi exploatează atât căutarea globală, cât și cea locală, și sunt des utilizați în domeniul ingineriei energetice.
7. Algoritmi genetici multi-obiectiv, AGMO – acești algoritmi sunt utilizați pentru a gestiona obiective multiple, de multe ori conflictuale. Cele mai populare exemple sunt NSGA-II (Non-dominated Sorting GA II) și SPEA2 (Strength Pareto Evolutionary Algorithm). Caracteristicile cheie sunt optimalitatea Pareto care este utilizată pentru selecție și mecanismele de conservare a diversității, cum ar fi distanța de aglomerare.
8. Algoritmi genetici codificați cu valori numerice (real-coded), AGR – acești AG folosesc numere reale în loc de șiruri binare, astfel sunt mai potriviți pentru probleme de optimizare continuă. Pentru operatorii genetici, la acești algoritmi sunt folosiți încrucișarea aritmetică și mutațiile neuniforme.

Dacă se ia în considerare doar strategia de execuție, atunci AG se împart în două mari clase, și anume modele secvețiale și paralele.

Modelele secvențiale folosesc o metodă secvențială de procesare a întregii populații, adică toți indivizii populației sunt evaluați pe rând. În consecință, la fiecare generație se verifică adaptarea indivizilor prin funcțiile corespunzătoare, se selectează părinții, se aplică reproducerea și mutația, iar apoi se înlocuiește populația cu cea nouă. Avantajele acestor modele reies din simplitatea cu care se implementează și faptul că sunt accesibile pentru probleme de dimensiuni mici sau medii. Deoarece, ele vor fi încete pentru probleme de dimensiuni mari și funcții de adaptare complexe. Modelele secvențiaele se împart în două mari categorii: modele statice și modele dinamice. Cele staționare se caracterizaeză prin faptul că toți parametrii rămân ficși, iar cele dinamice (adaptive) au parametrii variabili.

Modelele paralele sunt folosite pentru a crește viteza și diversitatea. Aceste modele se împart în trei mari ramuri (așa cum s-a menționat și anterior): mastre-slave, insulare și celulare.

AG master-slave sunt AG de tip centralizat, astfel, ele au o structură care suprinde un controler central, master, care gestionează populația. Acest modul central controlează mai multe unități, lucrătorii (slave) care evaluează într-un mod distribuit adaptarea cromozomilor. Principalul avantaj a acestor AG este ușurința cu care se implementează cu modificări minime de un AG secvențial.

AG insulari sunt algoritmi catalogați ca modele descentralizate, întrucât populația este împărțită în subpopulații (insule), iar fiecare insulă evoluează independent folosind propriul AG, dar ocazional, indivizii migrează între insule. Aceste modele de AG promovează diversitatea, ele fiind potrivite pentru sisteme distribuite. Adițional față de ceilalți AG, acești algoritmi au parametrii de migrare, precum frecvența, rata și topologia de migrare. Marele dezavantaj a AG insulari este necesitatea unei reglări atente a setărilor de migrare.

AG celulari au indivizii aranjați pe o grilă 1D sau 2D, iar operatorul de reproducere se realizează între indivizi din apropiere. Acești algoritmi oferă o diversitate mare a indivizilor, și sunt utile pentru rezolvarea problemelor spațiale. Principalele dezavantaje se referă la convergența mai lentă și implementarea mai complexă, comparativ cu alți AG. La fel ca AG insulari și aceste modele paralele sunt descentralizate.

O sinteză a modelelor secvețiale și parelele a algoritmilor genetici este prezentată în tabelul 10.1. Astfel, tabelul cuprinde o comparație între modelele secvențiale statice și dinamice, respectiv modelele paralele centralizate și descentralizate. În comparație s-au luat în considerare modul de execuție, structura populației, modul în adaptivitate și comunicarea în interiorul populației.

**Tabelul 10.1.** Modelele secvențiale și paralele de AG

| Modelul AG | Execuția | Populația | Adaptivitatea | Comunicare |
|---|---|---|---|---|
| Secvețial static | Un singur fir de calcul | Populație unică globală | Parametrii au valori fixe | Nu există |
| Secvențial dinamic | Un singur fir de calcul | Populație unică globală | Parametrii se adaptează, valori variabile | Nu există |
| Master-slave Paralel central | Mai multe direcții de calcul | Populație unică globală | De obicei comportament static | Centrală dintre master și slave |
| Insular Paralel descentralizat | Mai multe direcții de caclul distribuite | Mai multe populații | Comportament local, cu posibilitatea de a se modifica valorile | Periodică în timpul migrației |
| Celular Parallel descentralizat | Mai multe direcții de caclul distribuite | Populați sub forma de grilă | Comportament local, de obicei static | Locală între vecini |

Alte tipuri de AG sunt descrise sumar în următoarele paragrafe.

Algoritmii genetici Lamarckian sunt inspirați din evoluția lamarckiană în care indivizii pot să transmită genetic și trăsăturile învățate. Aceste modele genetice combină AG static cu căutarea locală, iar soluția îmbunătățită este codificată înapoi în genotip. Ele sunt folosite cu precădere în probleme de optimizare cu funcții de fitness costisitoare sau zgomotoase. Față de modelele darwiniene, la care se îmbunătățește doar adaptarea, la aceste modele, se îmnunătățește și genomul.

Algoritmii genetici Baldwinieni sunt similari celor precedenți, dar comparativ cu aceia, la acești algoritmi căutarea locală îmbunătățește adaptarea, dar nu modifică genotipul. Rezultatul acestei acțiuni este o adaptare mai bună, care rezultă într-o șansă mai mare de reproducere, dar trăsătura învățată nu este moștenită. Din punct de vedere biologic, acest modul genetic este mai precis decât cel Lamarkian.

Modelul algoritmului genetic cultural este un sistem pe două niveluri, care se compun dintr-o populație uzuală și un spațiu de credințe care ghidează populația pe baza cunoștințelor învățate. Beneficiu acestei abordări este faptul că se exploatează atât diversitatea populației, cât și cunoștințele culturale partajate.

Algoritmii genetici inspirați din calculul cuantic folosesc principii din calculul cuantic (de exemplu, superpoziție, qubiți) în codificare și operații. Caracteristica definitorie este reprezentarea probabilistică a indivizilor folosind biți cuantici. Avantajele aduse de aceste modele sunt diversitatea și convergența mai bune în unele probleme combinatorii.

Algoritmii genetici de nișă au drept scop menținerea mai multor soluții (nișe) în loc să convergă către una singură. Tehnicile folosite sunt partajarea adaptării, aglomerarea și speciația.

Modelul algoritmului genetic imunologic este inspirat din sistemele de imunitate artificiale. Astfel, sunt introduși operatori imuni precum clonarea, hipermutația și supresia. Marele avantaj este menținerea diversității și evitatea optimelor locale.

Algoritmii genetici coevolutivi cuprind populații multiple care evoluează interdependent. Aceste populații pot fi competitive sau cooperative.

Ultimul tip de algoritmi genetici este programarea genetică, unde în loc loc să evolueze soluții, evoluează programe sau expresii matematice. Astfel, sunt folosite structuri arborescente în loc de șiruri de caractere.

Dintre aceste tipuri numeroase de AG, cei care sunt folosiți des în electroenergetică și managementul energiei sunt modelele clasice, adaptive, elitiste, hibride, multiobiectiv, în timp real și paralele (în special insulare). În consecință, în capitolele următoare, unde sunt descriși operatorii genetic și celelalte elemente caracteristice AG se va pune accent doar pe aspectele care sunt în legătură cu acești AG.

## 10.5. Întrebări și exerciții

### Întrebări

1. Algoritmii genetici lucrează cu soluții, ci nu pentru soluții.
   Adevărat / Fals.

2. Cromozomii sunt codificările soluțiilor.
   Adevărat / Fals.

3. Adaptarea cromozomilor este cuantificată prin funcții de adaptare și obiectiv.
   Adevărat / Fals.

4. Algoritmii genetici sunt folosiți pentru a rezolva probleme de optimizare.
   Adevărat / Fals.

5. Termenii folosiți în AG sunt moșteniți din teoria evoluționistă.
   Adevărat/Fals.

# 11

# Algoritmi genetici
## Operatori genetici

Acest capitol descrie operațiile care stau la baza algoritmilor genetici, și anume operatorul selecție, reproducere și mutație. Înainte de acestea este prezentat modul de codificare a soluțiilor cu care lucrează algoritmii genetici.

### 11.1. Aspecte generale

După prezentarea evoluției istorice și a caracteristicilor fundamentale ale algoritmilor genetici din capitolul precedent, următorul pas este înțelegerea mecanismelor interne care determină puterea evolutivă a AG. Așa cum evoluția naturală se bazează pe codificarea informațiilor genetice, selecția determinată de supraviețuire, reproducerea prin încrucișare și mutația aleatorie, AG simulează aceste principii biologice printr-un set de operatori de bază. Astfel, se vor explora elementele constitutive esențiale ale unui algoritm genetic: codificarea (reprezentarea soluțiilor), selecția (modul în care indivizii sunt aleși pentru a se reproduce), reproducerea (mecanismele de încrucișare) și mutația (introducerea diversității). Acești operatori nu numai că definesc modul în care soluțiile candidate evoluează în timp, dar influențează și semnificativ eficiența și succesul algoritmului în găsirea unor soluții optime sau aproape optime.

Pornind de la o populaţie de cromozomi, aceşti operatori sunt folosiţi pentru crearea de noi cromozomi, adică noi soluţii posibile ale problemei de optimizare. Într-adevăr, pentru fiecare generaţie, selecţia asigură alegerea din întreaga populaţie a cromozomilor părinţi în vederea încrucişării acestora.

Operaţia de selecţie se face prin evaluarea gradului de adaptare a fiecărui cromozom la problema analizată. Cu cât funcţia de adaptare are o valoare mai mare, cu atât acel cromozom este mai adaptat. În consecinţă, cromozomii cei mai adaptaţi vor fi cel mai frecvent selectaţi în vederea încrucişării, iar genele lor vor

prolifera în generaţia următoare. După selectarea a doi cromozomi părinţi, aceştia participă la operaţia de încrucişare şi produc cromozomi urmaşi, fiecare dintre aceştia din urmă moştenind parţial genele părinţilor. Încrucişarea / reproducerea are rolul de a permite genelor „puternice", care se găsesc în cromozomi diferiţi să participe la formarea unor cromozomi – urmaşi, care să preia caracteristicile pozitive ale ambilor părinţi. Cromozomii nou creaţi sunt alteraţi prin schimbări de mică amploare la nivelul unor gene, cu ajutorul operatorului de mutaţie. Aceste schimbări asigură introducerea noutăţii la nivelul materialului genetic. O parte din aceste mutaţii vor produce cromozomi cu valori superioare ale funcţiei de adaptare. Altă parte dintre mutaţii poate dăuna procesului de căutare, în sensul în care noile fenotipuri sunt mai slabe. Acest proces de înrăutăţire locală a căutării afectează semnificativ evoluţia AG, deoarece noii cromozomi, cu funcţii de adaptare inferioare, vor fi eliminaţi în scurt timp de aplicarea operatorului de selecţie. Descrierea simplificată a modului de aplicare a operatorilor genetici – forma de principiu a unui AG.

Etapele funcționării unui algoritm genetic, care au fost descrise în mare anterior sunt sintetizate în figurile 11.1. și 11.2.

1 Formarea populaţie iniţiale $P(Gen)$, $Gen = 1$.
2 Evaluarea funcţiilor de adaptare pentru populaţia iniţială
3 Faza de evoluţie:

Repeat

3.1 Selectarea cromozomilor – părinţi din populaţia curentă $P(Gen)$.
3.2 Încrucişarea cromozomilor – părinţi din populaţia $P(Gen)$.
3.3 Aplicarea mutaţiilor cromozomilor – urmaşi din generaţia curentă.
3.4 Formarea populaţiei pentru noua generaţie $P(Gen+)$.
3.5 Evaluarea funcţiilor de adaptare pentru populaţia $P(Gen+)$.
3.6 Trecerea la o nouă generaţie $Gen = Gen + 1$.

until {criteriul de oprire}

4 Soluţia optimă corespunde cromozomului cu funcţia de adaptare maximă din ultima generaţie $P(Gen)$.

**Fig. 11.1.** Etapele funcționării unui AG

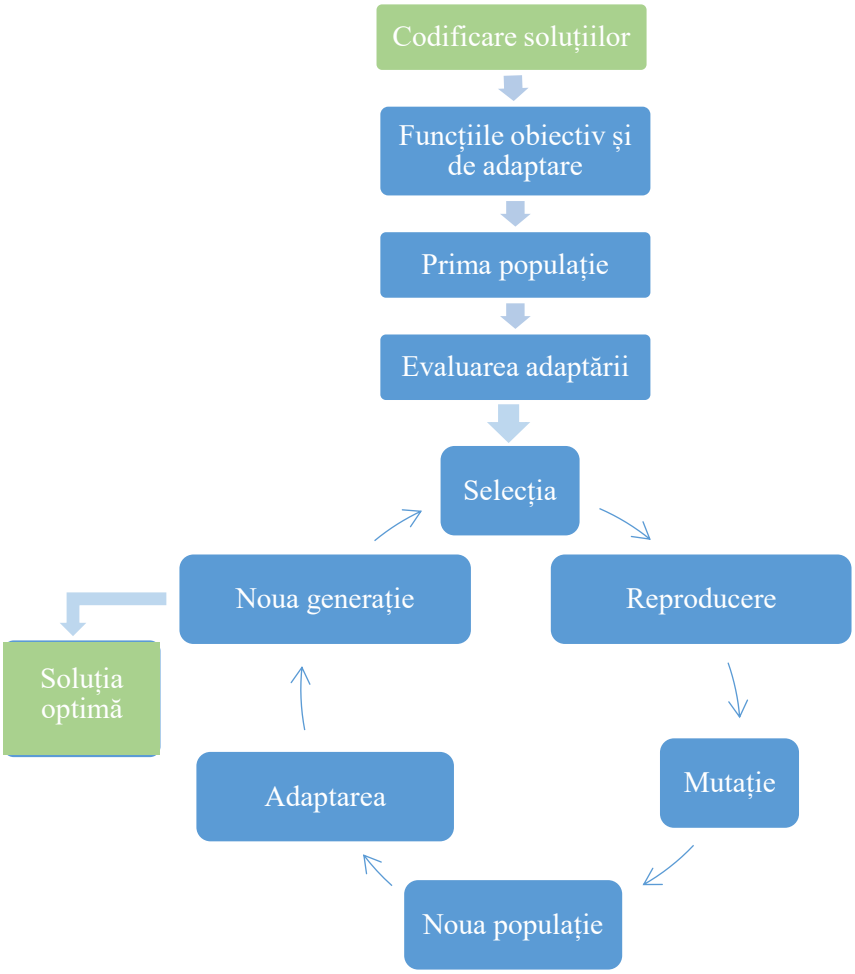



**Fig. 11.2.** Diagrama funcțională a unui AG

În următoarele subcapitole sunt descrise în detaliu aspectele care țin de codificarea (reprezentarea) soluțiilor, și operatorii genetici – selecție, reproducere și mutație.

### 11.2. Reprezentarea soluției

Reprezentarea soluției este o etapă critică în implementarea agloritmilor genetici, care presupune reprezentarea soluției sub forma unor lanțuri cromozomiale. Această acțiune este caracterizată prin următoarele aspecte: (1) posibilitarea definirii unei asemenea codificări, (2) realizarea unei reprezentări cât mai simple, dar în acelaşi timp corecte a unei soluţii generice a problemei analizate, și (3) cromozomul creat reprezintă o posibilă soluție.

Definirea corespunzătoare a soluţiilor sub forma lanţurilor cromozomiale presupune următoarele proprietăți care să fie atribuite codificării:

- reprezentarea să fie validă – fiecare cromozom rezultat în urma aplicării operatorilor genetici să poată fi decodificat sub forma unei soluţii.
- reprezentarea să acopere un domeniu suficient de larg din spaţiul soluţiilor posibile.
- reprezentarea să încurajeze deplasarea cromozomilor către soluţia problemei. Această proprietate presupune reprezentarea variabilelor corelate folosind gene învecinate în cadrul unui cromozom.

În concluzie, genele unui cromozom conţin codificarea valorilor parametrilor sau variabilelor care definesc funcţia de optimizat. Deci, codificarea presupune alegerea formatului de reprezentare pentru soluții (de exemplu, șiruri binare, vectori cu valori reale, permutări). În plus, este definit modul în care fiecare genă corespunde unei componente a soluției, respectiv aceste gene se combină pentru a forma un cromozom complet care reprezintă o soluție potențială.

În literatura de specialitate se cunosc mai multe modalităţi de codare a informaţiilor.

*Codificarea binară*

Folosirea codificării binare implică utilizarea pentru fiecare genă un șir de cifre binare. Acestea pot fi reprezentarea binară a unor valori numerice (zecimale), sau pot semnifica existenţa sau inexistenţa unei proprietăţi. Fig. 11.3 ilustrează structura a doi cromozomi rezultați prin codificarea binară.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| C1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| C2 | 0101 | 0110 | 0010 | 0010 | 0110 | 1010 | 0001 |

**Fig. 11.3.** Codificarea binară

În figura de mai sus se observă că genele iau valori binare singulare, C1 sau gru, C2. În cazul lui C1, valorile 0 și 1 pot semnifica prezenta sau absența unei caracteristici, de exeplmplu starea de funcțiune a unui generator sau receptor. În cazul lui C2, grupurile binare pot reprezenta forma binară a unor numere naturale, de exemplu puterea unui generator.

Acest tip de codificare, prin folosirea lui 0 și 1 este cea mai populară metodă de reprezentare a soluțiilor în cadrul AG, în special prin prisma faptului că în mod natural valorile cu care lucrează programele sunt binare.

Codificarea binară este potrivită pentru probleme caracterizate prin valori discrete, sau în care variabila decizională poate lua valori binare.

Avantajele codificării binare sunt simplitatea, întrucât șirurile binare sunt ușor de implementat și manipulat, uniformitatea, deoarece facilitează aplicarea operatorilor genetici precum încrucișarea și mutația, versatilitatea (potrivit pentru o gamă largă de probleme, în special cele cu variabile discrete) și eficiența, datorită faptului că poate reprezenta un spațiu de căutare mare în mod compact, permițând o explorare eficientă.

Contrar, limitările codificării binare țin de faptul că apar probleme de precizie, deoarece reprezentarea numerelor reale necesită discretizare, ceea ce poate duce la pierderi de precizie. În plus, complexitatea mapării este o altă problemă din cauză că, conversia între șiruri binare și variabile specifice problemei poate fi non-trivială. Scalabilitate este un punct slab dacă se lucrează cu problemele care necesită precizie ridicată, șirurile binare pot deveni excesiv de lungi. Și în final, există posibilitatea să apară redundanță, întrucât au apărut situații la unele reprezentări binare, care pot include soluții nevalide sau redundante, necesitând o manipulare suplimentară.

*Codificarea cu valori numerice (real-coded)*

Prin codificarea cu valori numerice, genele vor conţine direct valori, deci cromozomul va fi un şir de valori numerice reale. Codificarea cu valori este utilizată, de exemplu, pentru problema antrenamentului RNA, când trebuie aflată combinaţia de valori pentru ponderi care realizează eroarea minimă între ieşirile calculate şi cele impuse, conform unui criteriu dat, care va fi folosit pentru calculul gradului de adecvare.

Fig. 11.4 prezintă doi cromozomi, C3 și C4 care au fost codificați cu valori numerice. Se observă că, C3 folosește valori reale mai mici ca 10, iar C4 valori naturale cu două și trei cifre.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| C3 | 1,2 | 3,1 | 4,8 | 7,3 | 1,6 | 2,9 | 9,6 |
| C4 | 235 | 65 | 587 | 236 | 754 | 85 | 451 |

**Fig. 11.4.** Codificarea cu valori numerice

Acest tip de codificare a soluțiilor se recomandă să se folosească atunci când problema implică lucrul cu variabile continue, soluția necesită o precizie mare, reprezentarea naturală a soluției este non-binară sau nu poate fi codificată binar, iar procesul de codificare/decodificare este mult mai simplu, ușor de înțeles și accesibil.

Punctele forte are codificării prin valori numerice sunt:

- Reprezentare directă: Variabilele sunt reprezentate în forma lor naturală, ceea ce face interpretarea simplă.
- Precizie: Permite reprezentări de înaltă precizie fără limitările discretizării binare.
- Eficiență: Elimină costurile suplimentare de conversie între valori binare și reale, ceea ce poate spori eficiența computațională.
- Aplicabilitate: Potrivit pentru probleme de inginerie, optimizare și învățare automată unde parametrii sunt inerent continui.

Comparativ, punctele slabe ale codificării prin valori numerice rezultă din provocările care apar atunci când se lucrează cu acest tip de codificare. Cea mai semnificativă provocare este proiectarea operatorilor, deoarece operatorii genetici standard, cum ar fi încrucișarea și mutația, trebuie adaptați pentru date cu valori reale. De exemplu, se utilizează adesea tehnici precum Blend Crossover (BLX-α) sau Simulated Binary Crossover (SBX). Alte aspecte dificile țin de gestionarea constrângerilor și convergența prematură. Într-adevăr, este necesar să se asigure faptul că valorile mutate sau încrucișate rămân în limitele fezabile, ceea ce implică atenție sporită în implementare AG. Mai mult, fără mecanisme adecvate de conservare a diversității, algoritmul ar putea converge prematur către soluții suboptimale.

*Codificarea simbolică*

Codificarea simbolică presupune ca fiecare valoare a genelor conţine unul sau mai multe simboluri. Simbolurile pot fi caractere, litere, șiruri de litere sau operatori, fig. 11.5. Pentru problemele de permutare, simbolul poate fi numărul de ordine dintr-o secvenţă. Această abordare este deosebit de eficientă pentru problemele care implică manipulare simbolică, cum ar fi evoluția expresiilor matematice, a regulilor logice sau a structurilor de programe.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| C5 | A | B | B | C | A | D | E |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| C6 | X1y | Z3x | Y2x | Y1z | X2z | Z1x | X1z |

**Fig. 11.5.** Codificarea simbolică

În fig. 11.5 este prezentată codificarea simbolică folosind litere minuscule și majuscule alături de cifre, dar mulțimea simbolurilor folosite este mult mai largă. Astfel, simbolurile din interiorul genelor pot fi:

- Operatori – relații matematice precum +, -, *, /;
- Operanzi – variabile precum x, y, z, sau constante numerice (1, 3, 100,...).
- Funcții – funcții trigonometrice (sin, cos, ...) sau analitice (log, exp, ...).
- Structuri de control – operatori informatici precum ”Dacă”, ”Atunci”, ”Repetă / While”.

Codificarea simbolică este un instrument puternic în setul de instrumente al algoritmilor genetici, permițând evoluția unor soluții complexe și interpretabile în domenii în care reprezentarea simbolică este naturală și benefică.

Avantajele codificării simbolice sunt expresivitate (poate reprezenta structuri complexe, cum ar fi expresii matematice sau programe), flexibilitate (se adaptează ușor la diverse domenii problematice care necesită manipulare simbolică), și interpretabilitate (expresiile sau programele rezultate sunt adesea lizibile de către om, ajutând la înțelegere și analiză).

Precum celelalte tipuri de codificare, și codificarea simbolică prezintă provocări, care se referă la validitate (asigurarea faptului că operațiile genetice produc expresii valide și executabile), complexitate (gestionarea spațiului de căutare potențial vast al posibilelor combinații de simboluri) și evaluare (proiectarea funcțiilor de fitness adecvate pentru a evalua calitatea expresiilor simbolice).

*Codificarea cu arbori*

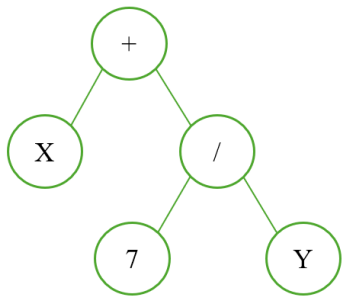
Codificarea bazată pe arbori este o metodă versatilă și expresivă în algoritmii genetici, permițând evoluția unor soluții ierarhice complexe. Aplicațiile AG care folosesc această codificare acoperă diverse domenii, de la regresia simbolică la generarea automată de programe, ceea ce o face un instrument valoros în domeniul calculului evolutiv.

Acest tip de codificare a soluțiilor, implică definirea unui cromozom, care este reprezentat ca un arbore de obiecte. Este utilizată în programarea genetică şi în programele evolutive. Cromozomul va conţine operatori, operanzi şi funcţii. Mai exact, fiecare cromozom este structurat ca un arbore, unde nodurile reprezintă funcții sau operații, iar frunzele (nodurile terminale) reprezintă intrări sau constante.

În codificarea bazată pe arbore, noduri funcționale sunt noduri interne care reprezintă operații sau funcții (de exemplu, +, -, *, /, sin, cos), iar nodurile terminale sunt noduri frunze care reprezintă variabile, constante sau intrări (de exemplu, x, y, 3.14).

Această reprezentare este potrivită în special pentru problemele în care soluțiile pot fi exprimate în mod natural ca structuri ierarhice, cum ar fi expresii matematice, reguli logice sau programe de calculator.

Fig. 11.6. ilustrează un exemplu de cromozom arbore, care folosește variabile (X și Y) și operatori (+ și /).




( + X ( / 7 Y ))

**Fig. 11.6.** Codificarea prin arbore

Punctele forte ale algoritmilor genetici care utilizează cromozomi codificați prin arbori sunt:

- Flexibilitatea - poate reprezenta o mare varietate de structuri, de la expresii simple la programe complexe.
- Modularitatea – sub-arborii pot reprezenta componente sau subrutine reutilizabile.
- Interpretabilitatea - soluțiile evoluate sunt adesea lizibile de către om, ajutând la înțelegere și analiză.

Provocările și aspectele care trebuie luate în considerare când sunt implicați cromozomi arbori, sunt umflare, complexitatea, și proiectarea operatorilor. Umflarea se referă la faptul că arborii pot crește excesiv de mari fără îmbunătățiri corespunzătoare ale fitness-ului, ceea ce duce la ineficiență. De asemenea, complexitatea este o problemă pentru că gestionarea și evaluarea arborilor mari poate fi intensivă din punct de vedere computațional. La fel ca în cazul codificărilor simbolice, proiectarea operatorilor este și aici o provocare, deoarece asigurarea faptului că operațiile genetice produc urmași validi și eficienți necesită o proiectare atentă. Pentru a atenua umplerea, se utilizează tehnici precum limitarea adâncimii arborelui, aplicarea presiunii parcimoniei (penalizarea arborilor mai mari) sau utilizarea metodelor de control al umplerii.

*Codificarea prin permutare sau ordine*

Codificarea prin permutare presupune ca fiecare cromozom este o permutare unică de elemente, asigurându-se că fiecare element apare exact o singură dată. Această metodă de reprezentare este adaptată pentru probleme în care soluția este în mod inerent o secvență ordonată, astfel, ea este deosebit de eficientă pentru problemele de optimizare combinatorie în care ordinea elementelor este crucială. O problemă șablon a acestui tip de codificare este problema curierului, care trebuie să aleagă ordinea de parcurgere a părților unui drum. Fig. 11.7. prezintă structura a doi cromozomi cu 7 gene.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| C7 | 1 | 3 | 2 | 7 | 4 | 6 | 5 |
| C8 | 2 | 4 | 1 | 3 | 7 | 5 | 6 |

**Fig. 11.7.** Codificarea prin permutare

Deci, codificarea prin permutare este esențială pentru menținerea unor soluții valide în problemele în care aranjamentul elementelor contează. Valorile genelor din cromozomi pot să fie cifre sau litere, caracteristic este că alelele nu se schimbă, ci doar se mută între gene.

Avantajele codificării prin permutări sunt:

- Reprezentarea naturală - modelează direct probleme în care soluția este o secvență ordonată.
- Conservarea constrângerilor - asigură că toate elementele sunt incluse exact o singură dată, menținând validitatea soluției.

- Aplicabilitatea - potrivit pentru o gamă largă de probleme de optimizare combinatorială.

Dezavantajele și provocările codificării prin permutări apar din cauza complexității operatorilor (proiectarea operatorilor de încrucișare și mutație care mențin permutări valide poate fi complicată), dimensiunea spațiului de căutare (numărul de permutări posibile crește factorial odată cu numărul de elemente, ducând la un spațiu de căutare vast), și convergența prematură (fără mecanisme adecvate de conservare a diversității, algoritmul ar putea converge către soluții suboptimale). Abordarea acestor provocări necesită o proiectare atentă a algoritmilor, inclusiv selectarea operatorilor genetici adecvați și a strategiilor de menținere a diversității.

Un alt aspect important în codificarea soluțiilor este poziția genelor în structura cromozomilor, care nu este aleatorie. De multe ori, între genele care sunt poziţionate în anumite locuri există interdependenţe neliniare complexe, iar pentru ca AG să fie eficient, evoluţia lui nu trebuie să se desfăşoare doar în sensul căutării unor alele cât mai adaptate, ci trebuie căutate seturi / blocuri de gene care împreună produc soluţia bună. Aceste secvenţe sunt descrise cu ajutorul noţiunii de schemă de gene.

O schemă reprezintă un tipar (șablon) de gene şi alelele lor pentru care unele alele au valori particulare, în timp ce altele sunt nedefinite. Schemele au fost definite în cazul reprezentării binare şi folosesc un alfabet format din trei simboluri, $\{0,1,*\}$:

- Simbolul „*” corespunde oricăreia din celelalte două valori
- Se spune că un lanţ cromozomial de lungime $L$, fie acesta $x = (x_1 x_2 \dots x_L)$ este o instanţă a schemei $S$, de aceeaşi lungime, fie acesta $S = (s_1 s_2 \dots s_L)$ , dacă pentru oricare dintre elementele schemei $s_i$ cu valoare specificată (0 sau 1) elementul cromozomului $x$ este identic cu aceasta: $x_i = s_i$.

J. Holland a definit două caracteristici care permit evaluarea efectelor operatorilor genetici asupra schemelor: ordinul şi lungimea specifică a schemei. Ordinul unei scheme, $O(S)$ este definit ca fiind numărul de elemente fixe din schema respectivă (elemente cu valori 0 şi 1). Lungimea specifică, notată cu $\Delta(S) = max(i|s_i \in \{0,1\}) - min(i|s_i \in \{0,1\})$ reprezintă distanţa maximă dintre elementele fixe (cu valori 0 sau 1) din schema respectivă.

Cele două măsuri definite pentru o schemă pot fi folosite pentru calculul probabilităţilor de supravieţuire a unei scheme ca urmare a aplicării diferiţilor operatori genetici. Probabilităţile de supravieţuire ale unei scheme *S* în urma aplicării mutaţiilor ($P_M$) şi încrucişărilor ($P_I$) sunt:

$$P_M = (1 - p_m)^{O(S)} \tag{11.1}$$

$$P_I = 1 - p_i \cdot \frac{\Delta(S)}{L-1} \cdot \left(1 - \frac{N(S)}{N}\right), \tag{11.2}$$

Unde $p_m$ și $p_i$ sunt ratele mutaţiilor şi încrucişărilor; $L$ este lungimea lanţului cromozomial; $N$ este numărul de cromozomi din populaţia curentă şi $N(S)$ este numărul de instanţe ale schemei $S$ printre cromozomii populaţiei curente.

### 11.3. Funcțiile obiectiv și de adaptare

O caracteristică de bază a algoritmilor genetici este reprezentată de funcțiile obiectiv și de adaptare.

Funcția obiectiv este funcția care definește scopul problemei, deci cuantifică cât de bună este o soluție în ceea ce privește cerințele problemei. În funcție de problemă, obiectivul poate fi minimizarea sau maximizarea acestei funcții. În consecință, funcția obiectiv cuantifică calitatea sau „adecvarea” unei soluții. În funcție de problemă, scopul poate fi maximizarea sau minimizarea acestei funcții. Exemplu de maximizare este problema - într-o problemă de optimizare a profitului, funcția obiectiv ar putea calcula profitul total, urmărind maximizarea acestei valori. Exemplu de minimizare este în ipoteza - într-o problemă de optimizare a rutei, funcția obiectiv ar putea calcula distanța totală parcursă, cu scopul de a o minimiza. Algoritmul genetic folosește funcția obiectiv pentru a evalua și compara soluții, selectând-o pe cele mai promițătoare pentru reproducere și explorare ulterioară.

Când se dezvoltă funcția obiectiv, se au în vedere următoarele:

1. Alinierea cu obiectivele – acest aspect implică faptul că funcția definită matematic, reflectă cu acuratețe obiectivele generale ale problemei.
2. Eficiență computațională - funcția trebuie să fie ușor de gestionat din punct de vedere computațional, deoarece va fi evaluată de numeroase ori în timpul execuției algoritmului.
3. Gestionarea constrângerilor – toate constrângerile problemei trebuie introduse, eventual prin termeni de penalizare care reduc adecvarea soluțiilor nefezabile.

4. Scalabilitate medie – funcția ar trebui să gestioneze cu eleganță diferite dimensiuni și complexități de intrare.

Funcția de fitness (adaptare) este utilizată de AG pentru a evalua și compara soluțiile. Aceasta, adesea derivă din funcția obiectiv, dar este adaptată pentru a se potrivi mecanismului de selecție al AG. De obicei, funcția de fitness transformă rezultatul funcției obiectiv într-o valoare non-negativă, unde valorile mai mari indică soluții mai bune. Această transformare este crucială deoarece multe metode de selecție din AG presupun că valorile de fitness mai mari corespund unor soluții mai bune.

Calitatea unui cromozom este apreciată prin gradul de adecvare, gradul în care cromozomul respectiv este eficient, adică gradul în care fenotipul corespunzător satisface problema. Acesta este dependent de funcţia obiectiv ce trebuie maximizată sau minimizată, și se realizează prin funcția de adaptare.

O situaţie deosebită apare în problemele în care este căutată o soluţie care să satisfacă un set de condiţii date. În acest caz, foarte mulţi cromozomi din spaţiul de căutare nu satisfac condiţiile cerute şi vor produce fenotipuri de valoare nulă, ceea ce reprezintă o problemă de selecţie. În această situaţie, este utilizată o funcţie de penalitate, a cărei valoare să indice gradul în care condiţiile au fost violate. O formă a funcţiei de penalitate este.

$$f_a(x) = f(x) + M^k \cdot w^T \cdot c_v(x) \tag{11.3}$$

Unde $w$ este un vector de factori de pondere nenegativi, $c_v$ este măsura violării oricărei costrângeri, $M$ este numărul generaţiei curent și $k$ este un exponent potrivit.

Mecanismul standard al AG foloseşte valori ale funcţiei de adaptare pozitive, iar procesul de optimizare urmăreşte maximizarea funcţiei de adaptare $F_A$, astfel că se impune scalarea funcţiei obiectiv la valori pozitive, în situaţiile în care aceasta încalcă condiţiile impuse.

În literatura de specialitate sunt propuse numeroase variante de scalare:

- scalarea prin inversare – se foloseşte numai atunci când problema constă în minimizarea funcţiei obiectiv $f$. În acest caz, pentru un cromozom $i$, funcţia de adaptare $F_{Ai}$ scalată se calculează ca inversa sumei dintre funcţia obiectiv $f_i$ şi o constantă $\theta$.

$$F_{Ai} = \frac{1}{f_i + \theta}. \tag{11.4}$$

- scalarea liniară dinamică – este o variantă a formei anterioare de scalare, care ţine seama şi de dinamica procesului de optimizare, prin alegerea

coeficientului a din relaţia în funcţie de valorile curente ale funcţiei – obiectiv. Transformarea liniară care se foloseşte în acest caz.

- scalarea după valoarea medie – în această situaţie se urmăreşte ca valorile medii ale funcţiilor – obiectiv şi de adaptare, pentru toţi cromozomii unei generaţii să fie egale.

$$F_{Ai} = a \cdot f_i + min(f_i) \tag{11.5}$$

În cazul operatorului de selecţie, se urmăreşte stabilirea numărului de cromozomi care conţin schema considerată după un anumit număr de generaţii, în care s-a aplicat numai operatorul de selecţie. În acest scop, se folosesc valoriile medii ale funcţiei de adaptare pentru toţi cromozomii populaţiei curente $F_A^{med}(t)$, respectiv pentru cromozomii populaţiei curente care conţin schema *S*, $F_A^{med}(S(t))$.

Prin combinarea celor trei operatori genetici (selecţia, încrucişarea şi mutaţia), se obţine teorema schemelor, care indică o estimare a numărului de cromozomi care conţin o anumită schemă, după parcurgerea unui anumit număr de generaţii. S-au definit şi alţi operatori şi micro-operatori genetici propuşi pentru a ajuta la implementarea algoritmilor genetici. Aceşti operatori fie au o aplicabilitate redusă, fie reprezintă doar propuneri care sugerează noi direcţii de dezvoltare a AG: operatorul de permutare, operatorul inversiune, operatori pentru modele cromozomiale complexe.

$$N(S(t+1)) \geq N(S(t)) \cdot \frac{F_A^{med}(S(t))}{F_A^{med}(t)} \cdot \left[1 - p_i \cdot \frac{\Delta(S(t))}{L-1} \cdot \left(1 - \frac{N(S(t))}{N}\right)\right] \cdot (1 - p_m)^{O(S(t))}$$

Cu privire la această teoremă se pot trage următoarele concluzii:

- Propagarea schemelor de la o generaţie la alta respectă teorema schemelor pentru dimensiuni mari ale populaţiei, *N*.
- În cadrul unei populaţii cu dimensiune finită, creşterea (descreşterea) exponenţială a numărului de instanţe ale unei scheme corespunde, fie completării totale a populaţiei cu instanţe ale acelei scheme, fie dispariţiei complete a schemei din populaţie, după un anumit număr de generaţii.
- Schemele caracterizate de o lungime specifică mare sunt păstrate cu o probabilitate mai mică decât cele cu lungime specifică mică.

O formă uzuală a funcției de fitness este

$$F(C) = \frac{1}{1+Fo(C)} \tag{11.6}$$

Unde $F(C)$ este funcția de adaptare, iar $Fo(C)$ este funcția obiectiv, $C$ este un cromozom.

Valorile funcțiilor de adaptare sunt pozitive și subunitare.

### 11.4. Operatorul selecție

Selecţia joacă un rol deosebit de important în evoluţia AG. Acest operator este folosit pentru alegerea unor cromozomi din populaţia curentă, care vor fi folosiţi pentru formarea unei noi generaţii, adică alegerea părinților. Toate metodele de selecţie cunoscute au la bază un mecanism tipic, ce foloseşte reguli probabilistice de „supravieţuire". Dacă în cazul sistemelor naturale supravieţuirea este determinată de capacitatea sistemului respectiv de a se adapta şi rezista în mediu, în cazul sistemelor artificiale care folosesc AG, supravieţuirea este legată strict de valoarea funcţiei de adaptare.

Selecţia cromozomilor-părinţi se face pe baza funcţiei de adaptare a fiecărui cromozom, astfel încât cromozomii cei mai bine adaptaţi vor avea cele mai mari şanse de a fi selectaţi. Luând în considerare că operatorul de selecţie se aplică secvenţial, cromozomii cu valori ridicate ale funcţiei de adaptare vor fi selectaţi de mai multe ori, însă şanse de selecţie existând şi pentru cromozomii mai puţin adaptaţi.

Selecţia se poate realiza în două moduri:

- prin reformarea întregii populaţii – se selectrază succesiv N cromozomi din populaţia curentă, care vor forma noua populaţie.
- prin selectarea a câte doi cromozomi-părinţi şi aplicarea operatorilor de încrucişare şi mutaţie pentru formarea cromozomilor-urmaşi. Această operaţie se repetă de N/2 ori, iar în final cromozomii-urmaşi înlocuiesc cromozomii-părinţi în totalitate, formând noua populaţie.

Cele mai folosite metode de selecţie sunt selecția uniformă, proporțională simplă, proporțională cu scalare, prin competiție, după modelul ruletei, după rang.

Selecţia uniformă – fiecare cromozom-părinte are şanse egale de a fi selectat, indiferent de valoarea funcţiei de adaptare asociate.

$$p_i = \frac{1}{N}, i = 1 \ldots N. \tag{11.7}$$

Selecţia proporţională simplă (fără scalare) – pentru fiecare cromozom se calculează probabilitatea de selectare luând în considerare aportul funcţia de adaptare în cadrul populaţiei. Dacă pentru populaţia curentă, fiecare cromozom $k$ are o funcţie de adaptare probabilitatea de selecţie a cromozomului $i$ se calculează ca raportul dintre propria funcţie de adaptare, probabilitatea de selecţie a cromozomului $i$ se calculează ca raportul dintre propria funcţie de adaptare şi suma valorilor funcţiilor de adaptare pentru întreaga populaţie.

$$p_i = \frac{F_{Ai}}{\sum_{k=1}^{N} F_{Ak}} \tag{11.8}$$

Selecţia proporţională cu scalare – este un caz particular al selecţiei proporţionale; corespunde situaţiei în care înaintea calculării probabilităţilor de selecţie, funcţia de adaptare este scalată.

Selecţia prin competiţie – se defineşte variabila $q$ – competiţie. Pentru selectarea unui cromozom, se aleg la întâmplare $q$ cromozomi din populaţia curentă, iar dintre aceştia se reţine cromozomul cu valoarea maximă a funcţiei de adaptare. Deşi, pentru acest tip de selecţie nu se determină probabilităţi de selecţie, se poate demonstra că probabilitatea de selecţie a cromozomului $i$ folosind tehnica $q$ – competiţiei este:

$$p_i = \frac{(N-i+1)^q-(N-i)^q}{N^q}. \tag{11.9}$$

Selecţia cu modelul ruletei – este o variantă a selecţiei proporţionale. Astfel, suma valorilor funcţiilor de adaptare pentru toţi cromozomii din populaţia curentă se asociază întregii lungimi a ruletei, care se împarte apoi în sectoare de lungimi egale cu proporţia funcţiei de adaptare a fiecărui cromozom în circumferinţa ruletei. Această proporţie reprezintă de fapt probabilitatea calculată cu relaţia.

Utilizarea în practică a acestei metode se realizează după modelul: pentru selectarea unui cromozom, se generează un număr aleatoriu în intervalul (0, 1) şi se identifică pe ruletă sectorul în care se încadrează acest număr. Cromozomul căruia îi corespunde acel sector va fi selectat.

Selecţia după rang – necesită evaluarea funcţiei de adaptare pentru toţi cromozomii şi ordonarea descrescătoare a acestora: cromozomul cel mai adaptat i se asociază rangul 1, iar cromozomul cel mai slab adaptat – rangul $N$. În continuare, probabilităţile de selecţie se calculează numai pe baza rangului fiecărui cromozom, fără a mai folosi funcţiile de adaptare. Utilizarea acestui tip de selecţie prezintă o serie de avantaje: scalarea din considerente de aducere a valorilor funcţiei obiectiv într-un anumit domeniu nu mai este necesară, metoda

de selecţie se poate aplica şi pentru probleme de maxim şi pentru probleme de minim, funcţia de adaptare poate fi identică cu funcţia obiectiv şi asigură accelerarea procesului de căutare.

În literatura de specialitate există mai multe relaţii pentru calculul probabilităţilor:

$$p_i = \frac{1}{N}\left[\eta - 2 \cdot (\eta - 1) \cdot \frac{i-1}{N-1}\right], \tag{11.10}$$

$$p_i = c \cdot (1-c)^{\eta-1}, \tag{11.11}$$

$$p_i = \frac{c}{1-(1-c)^N} \cdot (1-c)^{\eta-1}. \tag{11.12}$$

O altă metodă de selecție este metoda elitistă, prin care mereu cei mai adaptați dintre indivizi sunt selectați și transmiși în următoarea generație. Prin această abordare, se asigură faptul ca cea mai bună dintre soluții nu este pierdută ci se transmite la următoarea generație. Această metodă este adesea combinată cu metoda ruletei sau după rang.

Selecția prin trunchiere presupune ca doar primii x% dintre indivizi sunt selectați ca părinți. Este o metodă simplă și deterministă, dar poate duce la convergență prematură.

Eșantionare Stochastică Universală este versiunea îmbunătățită a selecție ruletei. Prin aceasta se aleg mai mulți părinți distanțați uniform pe distribuția de probabilitate. Avantajule acestei metode de selecție este reducerea erorii și a zgomotului de eșantionare.

Selecția Boltzmann este inspirată de recoacerea simulată, care folosește o probabilitate bazată pe temperatură pentru a favoriza indivizii cu fitness ridicat, permițând în același timp explorarea. Astfel, este utilă pentru evitarea minimelor locale în probleme puternic constrânse.

Selecția de nișă este utilizată în probleme de nișă/multimodale (de exemplu, planificarea restaurării cu configurații valide multiple). Această abordare implică ca indivizii apropiați unul de celălalt în spațiul soluției își împărtășesc fitness-ul, reducând dominanța unui grup de soluții.

### 11.5. Operatorul reproducere

Încrucişarea (reproducerea, împerecherea) este considerat ca fiind cel mai important operator folosit în cadrul AG. Încrucişarea a fost introdusă pentru a crea posibilitatea construirii unor noi cromozomi care să beneficieze de caracteristicile utile care se regăsesc în cromozomii ce formează populaţia

curentă. Transmiterea acestei informaţii se face prin schimbarea de segmente între cromozomi. Practic, se aleg doi cromozomi, denumiţi cromozomi-părinţi, care se recombină pentru a da naştere la doi noi cromozomi, denumiţi cromozomi-urmaşi, care trec în noua generaţie. Încrucişarea cromozomilor-părinţi se produce cu o probabilitate $p_I$, astfel încât este posibil ca cei doi cromozomi-părinţi să treacă în noua generaţie fără modificări. Pentru probabilitatea de încrucişare $p_I$ literatura de specialitate recomandă valori în limite relativ largi 0,6...0,95.

Încrucişarea aduce două efecte pozitive importante: reprezintă originea caracterului masiv paralel al calculelor şi asigură conservarea schemelor. În general, se folosesc următoarele tipuri de încrucişări: într-un punct, în două puncte, în *n* puncte şi uniformă. Încrucişarea într-un punct este forma tradiţională a operatorului de încrucişare. Pornind de la doi cromozomi-părinţi de lungime *L* – notaţia

$$a = (a_1 \dots a_{H-1} a_H a_{H+1} \dots a_L) \tag{11.13}$$

se generează un număr întreg aleatoriu H în intervalul $[1, L-1]$

$$b = (b_1 \dots b_{H-1} b_H b_{H+1} \dots b_L). \tag{11.14}$$

se schimbă segmentele finale ale celor doi cromozomi, începând cu gena *H*+1, fig. 11.8.

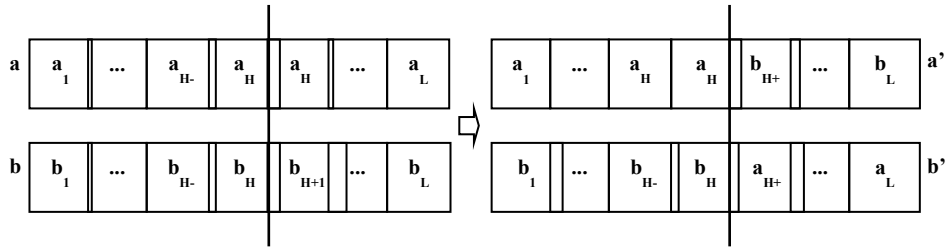



**Fig. 11.8.** Operatorul împerechere într-un punct

Încrucişarea în două puncte se desfăşoară astfel: după selectarea cromozomilor-părinţi, se generează numere aleatorii, iar schimbul între cromozomii-părinţi se face prin segmentele cuprinse între cele două numere.

Încrucişarea în două puncte se desfăşoară astfel: după selectarea cromozomilor-părinţi, se generează numere aleatorii, iar schimbul între cromozomii-părinţi se face prin segmentele cuprinse între cele două numere.

Generalizarea celor două procedee de încrucişare conduce la încrucişarea multi-punct sau în *n* puncte $H \in (0,1)$ , $H > {}^{1}/_{2}$.

În cazul încrucişării uniforme, pentru fiecare genă se generează un număr aleatoriu, cele două gene din cromozomii-părinţi se schimbă între ele:

$$a_i{}' = \begin{cases} a_i, H \leq {}^{1}/_{2} \\ b_i, H > {}^{1}/_{2} \end{cases}, \quad b_i{}' = \begin{cases} b_i, H \leq {}^{1}/_{2} \\ a_i, H > {}^{1}/_{2} \end{cases}. \tag{11.15}$$

Alte tipuri de încrucişare sunt: încrucişarea segmentată, încrucişarea prin amestecare, încrucişarea punctuală.

Metodele de încrucișare prezentate anterior sunt în special folosite pentru problemele în care s-a realizat codificarea binară. În cazul folosirii altor tipuri de codificare, se utilizează metodele prezentate în continuare.

Încrucișarea aritmetică este folosită în cazul codificării cu valori numerice. Prin această abordare, descendenții sunt combinații liniare ale părinților:

$$\text{Copil} = \alpha \cdot \text{Părinte}_1 + (1-\alpha) \cdot \text{Părinte}_2 \tag{11.16}$$

Încrucișarea euristică generează descendenți înclinați spre părintele mai potrivit. Această abordare funcționează numai dacă părinții sunt clasați și nu sunt egali. Dezavantajul este că poate duce la o convergență prematură.

Încrucișarea mixtă (BLX-α) a fost dezvoltată pentru modelele de AG cu valori reale. Genele descendenților sunt extrase dintr-un interval din jurul părinților. Astfel, dacă x1 și x2 sunt alele părinților, unde x1<x2, se determină distanța dintre ele

$$I = x1 - x2, \tag{11.17}$$

Valoarea genei copilului va fi aleasă din intervalul $[x1 - \alpha I, x2 + \alpha I]$, unde $\alpha$ este un parametru care controlează distanța față de valorile genelor părinților. O variantă pentru determinarea valorii genei copilului ($x_{\text{copil}}$) este să se aleagă o valoare aleatoare, *r*, în intervalul [0, 1], și folosirea relației (11.18), pentru determinarea acesteia.

$$x_{copil} = (x2 - x1)(r \cdot (1 + 2\alpha) - \alpha) + x1, \tag{11.18}$$

Încrucișarea binară simulată (SBX) imită comportamentul încrucișării binare în AG care folosește codificarea cu valori numerice. Această metodă este populară în optimizarea sistemelor de alimentare multi-obiectiv (de exemplu, NSGA-II). Astfel, dacă x1 și x2 sunt alele părinților, unde x1<x2, se generează factorul $\beta_q$, care controlează cât de departe de părinți va fi urmașul.

$$\beta_q = \begin{cases} (2u)^{\frac{1}{\eta_c+1}}, dacă\ u \leq 0{,}5 \\ \left[\frac{1}{2(1-u)}\right]^{\frac{1}{\eta_c+1}},\ dacă\ u \geq 0{,}5. \end{cases} \quad (11.19)$$

Unde $\eta_c > 0$ este indexul de distribuție a operatorului de încrucișare, u este o valoare aleatorie în intervalul (0, 1).

Următorul pas este determinarea celor doi copii (valorilor genei alese) ai cromozomilor părinți, care satisfac condiția de a se afla într-un interval predefinit:

$$C_1 = 0{,}5 \cdot \left[\left(1 + \beta_q\right)x1 + \left(1 - \beta_q\right)x2\right], \quad (11.20)$$

$$C_2 = 0{,}5 \cdot \left[\left(1 - \beta_q\right)x1 + \left(1 + \beta_q\right)x2\right] \quad (11.21)$$

Sau valori mai simetrice

$$C_1 = 0{,}5 \cdot \left[x1 + x2 - \beta_q(x2 - x1)\right], \quad (11.22)$$

$$C_2 = 0{,}5 \cdot \left[x1 + x2 + \beta_q(x2 - x1)\right] \quad (11.23)$$

Încrucișarea de ordine (Order Crossover (OX)) este utilizat pentru probleme bazate pe permutări, cum ar fi angajarea unităților sau planificarea. Această abordare asigură o secvență validă fără duplicate. Dacă se presupune selecția a doi părinți cromozomi, în prima etapă a folosirii acestui operator de încrucișare se aleg două puncte, iar valorile genelor dintre aceste punte se vor folosi neschimbate de la primul cromozom și transmise mai departe către copil. Apoi, de la părintele doi se preiua genele, însă se sare peste valorile care există deja la copil. Astfel, se asigură faptul că nu vor fi valori care se repetă.

Încrucișarea mapată parțial (Partial Mapped Crossover (PMX)) este destinată utilizării codificării prin permutări; aceasta păstează mai multe informații poziționale decât OX. Modul de operare a acestui tip de încrucișare este următorul:

1. Selecția celor doi părinți și a două puncte pentru încrucișare;
2. În corpul cromozomului copil se copiază segmentul dintre cele două puncte de la primul părinte;
3. Se crează mapări (legături) între segmentele din mijloc, astfel încât să se rețină valorile corespondente înlocuite. Aceste asocieri va ajuta în rezolvarea conflictelor.

4. Se completează genele rămase libere cu valorile de la părintele 2, și se vor folosi mapările de la punctul 3, în cazul în care apar conflicte, adică valori care se dublează.

La final, copilul va păstra ordinea valorilor luate de la părinți, iar fiecare valoare apare o singură dată. Această metodă este mai rapidă decât precedenta.

În cazul în care se folosește codificarea prin arbori, cel mai populară metodă de încrucișare este prin subarbori, în care cromozomul copil va cuprinde subarbori de la cei doi cromozomi părinți.

### 11.6. Operatorul mutație

Mutaţiile se referă la schimbări cu caracter aleatoriu şi de mică amploare care se produc la nivelul unui singur cromozom.

Operatorul mutaţie a fost introdus de Holland ca un operator de „fundal", care din când în când modifică alele izolate din cromozom, prin bascularea între 0 şi 1 sau reciproc. Principalul rol al mutaţiilor este cel de a asigura „materia primă" pentru operatorul încrucişare, astfel încât acesta să acţioneze asupra unui domeniu cât mai complet din spaţiul de definiţie. Aplicarea mutaţiilor reprezintă o asigurare împotriva „blocăii" AG în extreme locale.

Prin analogie cu modelul selecției naturale, mutaţiile se produc cu o anumită probabilitate $p_m \in (0,1)$ denumită rata mutaţiilor, care de obicei este foarte mică. Valorile recomandate în literatura de specialitate sunt $p_m = 0{,}001\ldots0{,}01$. În numeroase cazuri practice valorile sunt mult mai mari, care duc la rezultate mai bune. În practică aplicarea operatorului de mutație se face prin selectarea la întâmplare a unei gene din cromozomul-urmaș şi generarea unui număr aleatoriu $h \in (0,1)$.

Dacă $h \leq p_m$ se produce bascularea valorii genei respective între 1 şi 0 sau reciproc; în caz contrar, când $h \geq p_m$ gena rămâne neschimbată.

Din punct de vedere formal, pentru un cromozom

$$x = (x_1 x_2 \ldots x_L) \tag{11.16}$$

$$x_i{}' = \begin{cases} 1 - a_i, h \leq p_m \\ b_i, h > p_m \end{cases} \tag{11.17}$$

În practică sunt frecvent utilizate următoarele tipuri de mutație: mutația prin inversiune şi mutaţia prin reinițializare. Mutaţia prin inversiune este cea propusă iniţial de către Holland, însă acest tip de mutaţie poate fi folosită doar în cazul reprezentării binare.

Mutaţia prin reiniţializare se aplică în cazul reprezentărilor numerice (întregi sau reale) şi constă în schimbarea valorii asociate unei gene în mod aleatoriu, conform domeniului de definiţie corespunzător $[a_i, b_i]$. Simpla reiniţializare a unei gene prezintă însă un neajuns important în raport cu mutaţia prin inversiune. Din acest motiv, în practică mutaţia prin reiniţializare foloseşte un alt mecanism, care ajustează valoarea curentă a genei prin adăugarea sau scăderea unei valori mici. În acest caz sunt mai multe variante de implementare. Cea mai simplă este mutaţia cu pas fix – foloseşte o valoare constantă a mărimii $\delta$.

În cazul mutaţiei uniforme, corecţia $\delta$ nu mai este constantă, ci ia valori într-un interval. Mutaţia gaussiană generează corecţia $\delta$ pentru fiecare genă, folosind distribuţia Gauss de valoare medie 0 şi dispersie precizată $\sigma$.

$$[\delta_{min}, \delta_{max}] \tag{11.18}$$

Tuturor tipurilor de mutaţie li se poate suprapune o procedură de adaptare în timp a corecţiei $\delta$ astfel încât aceasta să se micşoreze pe măsură ce algoritmul genetic se apropie de convergenţă. Alte tipuri de mutaţii folosite în cazul reprezentării prin numere întregi sunt mutaţia la limită şi mutaţia neuniformă.

## 11.7. Întrebări și exerciții

### Întrebări

1. Codificarea soluției se poate face prin folosirea unor nume reale
   Adevărat / Fals.

2. Codificarea soluției se referă la reprezentarea soluției problemei sub forma unro lanțuri de valori cromozomiale.
   Adevărat / Fals.

3. Funcțiile obiectiv și de adaptare definesc problema de rezolvat și se referă la două aspecte fără legătură între ele.
   Adevărat / Fals.

4. Operatorul de selecție se folosește pentru a slecta copii cromozomi care îi vor înlocui pe cromozomi părinți.
   Adevărat / Fals.

5. Operatorul de reproducere sau încrucișare are rolul de a obține noi populații prin încrucișarea a două populații anterioare.
   Adevărat / Fals.

6. Operatorul de mutație se folosește pentru a crea noi soluții care nu au nici o legătură cu soluțiile precedente.
   Adevărat / Fals.

**Exerciții**

1. Să se scrie câte un exemplu din domeniul managementului energiei prin implicarea tuturor metodelor de codificare a soluției.

2. Să se folosească operatorul de reproducere pentru diferite tipuri de AG.

3. Să se folosească operatorul de mutație pentru diferite tipuri de AG.

# 12

# Algoritmi genetici
## Populații și generații

Acest capitol se focalizează pe prezentarea modului în care se formează și modifică populațiile de indivizi în cadrul AG, respectiv a caracteristicilor generațiilor de soluții. Acest capitol cuprinde și criteriile de oprire a algoritmilor genetici, respectiv factorii care influențează structura și funcționarea AG.

### 12.1. Introducere

În centrul fiecărui algoritm genetic se află două concepte cheie: populația, care reprezintă diversitatea soluțiilor explorate, și generația, care marchează procesul iterativ al evoluției. Calitatea și diversitatea populației, împreună cu numărul de generații, influențează direct capacitatea algoritmului de a explora spațiul de căutare și de a evita convergența prematură. Pe lângă aceste elemente structurale, mai mulți factori influențează comportamentul și performanța unui algoritm genetic. Aceștia includ alegerea strategiei de selecție, operatorii de reproducere și mutație, metodele de codificare și criteriile de oprire. Înțelegerea modului în care fiecare dintre aceste componente interacționează este crucială pentru proiectarea unor algoritmi genetici eficienți și eficace.

Astfel, dacă capitolul precedent a prezentat principiile de bază ale AG, și anume codificarea soluției, funcțiile oviectiv și fitness și operatorii genetici, care oferă o imagine microscopică asupra soluțiilor, întrucât lucrează cu indivizii, acest capitol se focalizează pe cele două elemente macroscopice ale AG, și anume populația și generația.

Mai mult, capitolul enumeră detaliat factorii care influențează eficiența AG în procesul de găsire a soluției optime. Finalul capitolului cuprinde informații despre metodele de implementare a algoritmilor genetici.

## 12.2. Populații

Formarea unei noi populaţii sau trecerea la o nouă populație este o operaţie foarte importantă în cadrul AG. În lucrul cu algoritmii genetici, prima populație este generată aleator.

Prima populație denumită și populația inițială este componenta fiecărui AG, care declanșează întregul proces evolutiv. Mai exact, prima populație este un set de soluții (cromozomi / indivizi) potențiale care sunt generate aleatoriu sau euristic la începutul AG. Fiecare individ codifică o posibilă soluție la problema de rezolvat.

Prima populație este caracterizată prin următoarele caracteristici:

- Generare aleatorie sau semi-aleatorie - Cel mai frecvent, prima populație este generată aleatoriu pentru a asigura diversitatea și a reduce erorile. În unele cazuri, se poate însămânța populația cu soluții bune cunoscute sau euristici pentru a ghida căutarea.
- Dimensiune - Numărul de indivizi este definit de parametrul dimensiunii populației, care va rămâne fix pe parcursul funcționării AG. Dimensiunile tipice variază de la 20 la câteva sute, în funcție de complexitatea problemei.
- Reprezentare - Fiecare individ este de obicei codificat ca un șir (binar, cu valori reale sau de alte tipuri) care reprezintă variabilele de decizie.

Alegerea primei populații este foarte importantă, deoarece afectează diversitatea, comportamentul de convergență și procesele de explorare și exploatare. Într-adevăr, o primă populație diversă explorează mai mult din spațiul de căutare, crescând șansa de a găsi optimul global. Mai mult, o primă populație slabă poate duce la o convergență prematură - algoritmul s-ar putea bloca în optimele locale, iar răspândirea inițială ajută la echilibrul dintre explorarea spațiului de soluții și exploatarea ulterioară a soluțiilor bune cunoscute.

Există mai multe metode de generare a primei populații: inițializarea pur aleatorie, inițializarea euristică, inițializarea bazată pe opoziție și eșantionarea supercub latină.

Cea mai populară, simplă și comună metodă este inițializarea pur aleatorie, în care valorile sunt eșantionate uniform în intervalul permis. Astfel, dacă se folosește codificarea binară, atunci genele cromozomilor populației vor lua valori de 0 și 1 într-un mod aleator. În cazul codificării cu valori numerice, se vor folosi valori pentru gene într-un interval predefinit.

Inițializarea euristică presupune includerea de indivizi generați folosind cunoștințe de domeniu. Această metodă este utilă în probleme de inginerie (cum ar fi sistemele energetice).

Inițializarea bazată pe opoziție generează atât un candidat, cât și opusul său pentru a crește diversitatea.

Eșantionarea hipercub latină este folosită pentru populații a AG care au cromozomi cu valori numerice, ea asigură proprietăți mai bune de umplere a spațiului.

Indiferent de tipul populației, și anume, prima populație sau o nouă populație oarecare, aceasta se poate caracteriza prin următoarele atribute:

- Structură – algoritmul cuprinde indivizii, care se caracterizează prin diferite forme dedicate problemei de rezolvat.
- Dimensiune – trăsătură crucială a AG, care influențează convergența algoritmului, întrucât o populație mică crește rapiditatea, dar poate conduce la o convergență prematură, iar o populație mare dă o diversitatea mai mare și soluționare robustă, dar cu costul suplimentar a relațiilor matematice și timpului.
- Diversitate – menținerea diversității unei populații este esențială în evitatea blocării algoritmului în optime locale. Menținerea sau creșterea diversității unei populații se obține prin operatorii genetici precum, mutația și migrația.
- Evoluție – cu fiecare generație, populația unui AG evoluează prin implicarea operatorilor genetici de selecție, încrucișare, mutație și înlocuirea indivizilor din vechea populație.
- Tip – un algoritm genetic lucrează cu un anumit tip de populație, care poate fi generațională, statică sau insulară. În cazul populațiilor generaționale, o populație veche se pierde total (întrucât ea este pentru o singură generație), ea fiind înlocuită de noi indivizi, astfel se pot pierde soluții bune, dacă nu se folosește selecția elitistă. La o populație statică se înlocuiesc doar unii dintre indivizi, restul trecând în generația următoare. O populație bazată pe insule cuprinde subpopulații care evoluează individual, astfel se păstrează o diversitate mărită și rezultate bune, în special prin operatorul de migrație.
- Convergență – o populație se spune că a ajuns convergența când majoritatea soluților sunt similare. Acest lucru poate însemna, că algoritmul este aproape de optimul global, sau s-a blocat la un optim local.

Sunt populații hibride, care au mai multe funcții de adaptare și codificări, ele fiind folosite în special în probleme multi-obiectiv și sisteme complexe care trebuie optimizate.

Pornind de la populaţia existentă formată din cromozomii-părinţi, aplicarea operatorilor de încrucişare şi mutaţie duce la formarea de cromozomi-urmaşi. Faptul că AG lucrează cu populaţii de dimensiune fixă, apare problema cum cromozomii-urmaşi îi înlocuiesc pe cromozomii-părinţi. Există două tehnici de bază pentru formarea unei noi populaţii: înlocuirea completă sau înlocuirea selectivă.

Primul tip presupune că noua generaţie este formată exclusiv din cromozomi-urmaşi, pierzând orice informaţie directă de la părinţi.

Pentru înlocuirea selectivă, cromozomii-urmaşi sunt comparaţi cu părinţii şi, pe baza unui criteriu, se decide care dintre cromozomi trec în generaţia următoare.

Plecând de la aceste două tehnici de bază au fost derivate numeroase variante, înlocuirea la fiecare generaţie a unui singur cromozom sau a unui număr finit de cromozomi, stabilit de regulă prin generarea unui număr aleatoriu între 1 şi $N$ (numărul de cromozomi din populaţia curentă).

O procedură specială folosită de aproape toate tehnicile de înlocuire este elitismul. Prin această procedură se realizează o copie a celui mai adaptat cromozom din generaţia curentă în generaţia următoare. Deşi procedura este foarte simplă, elitismul se dovedeşte ca fiind foarte eficient.

O altă variantă de înlocuire selectivă constă în compararea cromozomului-urmaş cu un părinte ales la întâmplare din populaţia curentă şi îl înlocuieşte pe cel din urmă dacă are o funcţie de adaptare mai mare

O extindere a acestei tehnici este înlocuirea prin competiţie: se selectează $q$ cromozomi-părinţi din populaţia curentă şi cel mai slab dintre aceştia este înlocuit de cromozomul-urmaş; se selectează $q$ cromozomi-părinţi din populaţia curentă, dar dintre aceştia va fi înlocuit cel care se aseamănă cel mai bine cu urmaşul său.

O ultimă tehnică este înlocuirea prin călire simulată – cromozomii-urmaşi cei mai bine adaptaţi îi vor înlocui pe părinţii mai slabi.

Pentru evitarea extremelor locale, în unele situaţii este permisă înlocuirea unor părinţi mai bine adaptaţi.

Diversitatea populațiilor este un parametru critic în eficiența procesului de optimizare. Dacă diversitatea scade prea repede, atunci există riscul ca AG să se blocheze la optime locale. În consecință, diversitatea trebuie monitorizată și

menținută pentru a crește eficiența AG. Pentru această acțiune, sunt folosiți indici ai diversității; cei mai comuni sunt diversitatea genotipică, diversitatea fenotipică și măsuri bazate pe entropie.

Diversitatea genotipică se referă la indicatori care cuantifică diferențele dintre genele unor indivizi. Pentru indivizi reprezentați binar, se folosește cel mai adesea distanța Hamming ($d_H$):

$$D = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_H(x_i, x_j) \tag{12.1}$$

$$d_H(x_i, x_j) = \sum_{k=1}^{p} |x_{i,k} - x_{j,k}| \tag{12.2}$$

Unde $x_i$ și $x_j$ sunt cei doi cromozomi care se compară, iar $k$ este indicele genei.

Pentru cromozomii care folosesc valori numerice se utilizează distanța Euclidiană:

$$D = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \|x_i - x_j\| \tag{12.3}$$

Diversitatea fenotipică este legată de diferența dintre valorile funcțiilor de adaptare, ci nu dintre valorile genelor. Cea mai simplă relație matematică pentru a cuantifica diferența este:

$$\sigma_f = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(f_i - \bar{f}\right)^2} \tag{12.4}$$

Unde $\bar{f}$ este valoarea funcției de adaptare față de care se determină distanța.

Măsurile bazate pe entropie presupun determinarea entropiei pentru fiecare genă a cromozomilor, de exemplu folosind relația lui Shannon, și apoi analiza acesteia. Dacă entropia este mare atunci rezultă o diversitate mare.

Dimensiunea populațiilor poate să difere de la o generație la alta, ea fiind ajustată în funcție de necesități. Astfel, se va crește eficiența și adaptivitatea AG.

Modificarea dimensiunii unei populații se poate realiza prin trei abordări: redimensionare dinamică, adaptarea bazată pe funcția fitness și modelele bazate pe vechime.

Redimensionarea dinamică presupune micșorarea sau mărirea numărului de indivizi ai populație în funcție de etapa în procesul evolutiv și starea populație. În consecință, se începe cu populații de dimensiuni mici, care apoi vor crește în mărime dacă algoritmul dă semne de stagnare. La finalul procesului de optimizare, când apar semne de convergență, se reduce dimensiunea populației pentru a accelera exploatarea.

Adaptarea bazată pe funcția fitness se realizează astfel: dacă diversitatea valorilor funcției fitness este scăzută, se crește dimensiunea populației pentru a introduce mai mulți indivizi, și astfel a intensifica procesul de explorare.

În cazul folosirii modelelor de populație bazate pe vechime, indivizilor li se asociază un indicator de vechime, iar în timpul procesului evolutiv, dimensiunea populației se va modifica, întrucât indivizii bătrâni mor, fiind înlocuiți de alții noi. Această abordare este utilă pentru menținerea diversității.

Numărul de indivizi care sunt adăugați sau eliminați nu este fix, și de altfel nu sunt reguli clare legat de acesta, dar atunci când dimensiunea populației este adaptivă, atunci se recomandă să se ia în considerare următoarele:

- Numărul de indivizi înlocuiți este de 5-20%. În situații mai drastice, în care apare o stagnare evidentă, se pot înlocui până la 50% dintre indivizi.
- Adăugarea de indivizi în zonele mai puțin aglomerate, și eliminarea de indivizi din zonele aglomerate (10-30% dintre indivizii similari) pentru a evita optime locale.
- Redimensionarea este declanșată doar atunci când anumite condiții sunt îndeplinite, de exemplu indicele de diversitate scade sub o anumită valoare impusă.
- În cazul aplicațiilor cu resurse limitate, ajustarea dimensiunii populației se va face în funcție de resursele energetice sau computaționale.

Managementul populației în cazul algoritmilor multi-obiectiv este diferit întrucât nu se caută o singură soluție globală, ci un set de soluții Pareto-optimale (o soluție Pareto-optimă este o soluție în care nici un obiectiv nu poate fi îmbunătățit fără o deteriorare a altui obiectiv, adică este compromisul care se face pentru a satisface toate obiectivele).

De exemplu, în cazul AG de tip NSGA-II, se folosește sortarea nedominantă (indivizii sunt clasificați în fronturi pe baza dominanței Pareto), distanța de aglomerare (se măsoară densitatea în jurul fiecărei soluții pentru a menține diversitatea) și elitismul (se combină populațiile părinte și urmași, se selectează cea mai bună variantă folosind rangul și diversitatea). Astfel, caracteristicile cheie ale populațiilor algoritmilor NSGA-II sunt: (1) distanța de aglomerare, prin care se încurajează fronturile Pareto dispersate și indivizii din regiunile mai puțin aglomerate sunt favorizați; (2) conservarea frontului Pareto, prin care algoritmul menține un set divers de compromisuri între obiective; (3) arhivarea populației – stochează o arhivă de soluții nedominante, și ajută la utilizarea arhivelor externe în probleme cu mai multe obiective sau dinamice.

### 12.3. Generații

La algoritmii genetici, generația este termenul care cuprinde un ciclu evolutiv, adică începând de la selecție, la încrucișare și până la crearea unei noi populații. Fiecare generație duce algoritmul spre soluția optimă globală, deoarece în fiecare generație se explorează într-un mod inteligent spațiul soluțiilor, prin luarea în considerare a trecutului informațional, și se folosește constrângerea selectivă pentru a îmbunătăți adaptarea soluțiilor în timp.

Procesul operațional al unui generații este ilustrat în fig. 12.1, unde se pot identifica cele patru etape mari ale unei generații. Însă, trebuie specificat că, acest proces pornește cu prima populație, și se încheie când este atins criteriul de oprire.

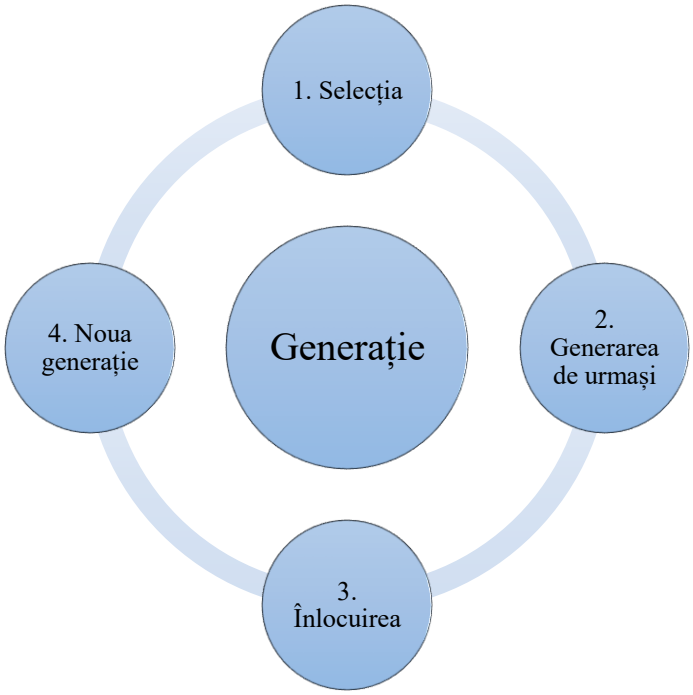



**Fig. 12.1.** Diagrama unei generații la AG

Prima populație care este componentă a generației inițiale are un impact mare asupra eficienței AG, deoarece influențează convergența acestuia. Prima populație trebuie să cuprindă o mare diversitate de indivizi pentru a mări spațiul de căutare. Mai multe despre acest subiect se găsesc în capitolul 11.

O nouă generație se formează astfel:

1. Se evaluează populația curentă pentru a determina adaptarea indivizilor. Așa cum s-a specificat și înainte, prin valoarea adaptării se determină care indivizi sunt fiabili pentru a deveni părinți, adică sunt cei mai adaptați.
2. Generarea de noi indivizi cuprinde trei pași:
    a. Selecția acelor indivizi care vor fi folosiți pentru a produce urmași.
    b. Realizarea încrucișării pentru a obține noi cromozomi.
    c. Implicarea mutației cu scopul de a obține noi cromozomi mai diverși
3. Înlocuirea părinților, care poate fi integrală sau parțială. S-a evidențiat în subcapitolul precedent că această acțiune poate duce la înlocuirea integrală (generațională), parțială sau elitistă.
4. Noua populație obținută merge la următoarea generație.

Criteriul de oprire este aspectul care infleunțează procesul computațional al AG, întrucât, algoritmul va repeta ciclul menționat anterior până când criteriul de oprire este îndeplinit.

Criteriul de oprire cel mai des folosit în cazul AG este numărul maxim de generaţii (iteraţii), impus a priori. În unele situaţii algoritmul genetic poate fi încheiat atunci când se ajunge la soluţia dorită - atingerea unei valori prag a funcției de adaptare impuse și lipsa unei îmbunătăţiri semnificative a soluţiei optime timp de un număr dat de generaţii, și diversitatea populației a scăzut sub un prag impus.

Cel mai relevant criteriu de oprire este atunci când populaţia a convers către o soluţie acceptabilă. Astfe, dacă este o populație de cromozomi

$$x_i = (x_{i1}x_{i2}\dots x_{iL}), i = 1, \dots, N. \tag{12.5}$$

Unde $x$ reprezintă populația, cu cei $N$ cromozomi ai populaţiei, $x_{ij}$ reprezintă alela / valoarea genei $j$ a cromozomului $i$.

Relaţia se poate aplica numai în cazul reprezentărilor binare şi se verifică uşor că factorul de diversitate satisface condiţia:

$$0{,}5 \leq D(t) \leq 1{,}0. \tag{12.6}$$

Cu cât cromozomii diferă mai mult între ei, cu atât gradul de diversitate este mai mic şi invers: cromozomii cu o structură foarte asemănătoare determină un grad de diversitate mai mare.

Evaluarea convergenţei populaţii se face folosind factorul de diversiune a populaţiei, care măsoară abaterea maximă între cromozomi după relaţia matematică:

$$D(t) = \frac{1}{L \cdot N} \sum_{j=1}^{L} max\left[\sum_{i=1}^{N}(1 - x_{ij}) \cdot \sum_{i=1}^{N} x_{ij}\right] \qquad (12.7)$$

Utilizarea factorului de diversitate drept criteriu de oprire presupune definirea unei valori de prag :

$$D_{max} \approx 0{,}95 \cdots 0{,}99 \qquad (12.8)$$

şi întreruperea algoritmului în momentul în care

$$D(t) \geq D_{max}. \qquad (12.9)$$

Un alt aspect important este evaluarea generațiilor. Indicatorii metrici ai generațiilor se bazează pe: cea mai bună valoare a funcției fitness, valoarea medie a funcției fitness, diversitatea (deviația standard a condiției fizice sau a genotipurilor), și rata de convergență. Acești indicatori se folosesc pentru a determina starea unei generații, precum diagnosticarea convergenței premature, sau reglarea parametrilor algoritmului sau vizualizarea progresului AG.

La final se poate afirma faptul că, un AG nu are garanția că converge către optimul global, dar pe parcursul a unui număr infinit de generații, poate aproxima soluțiile optime (conform teoremei schemei și ipotezei blocurilor de construcție).

### 12.4. Factorii care influențează AG

După realizarea unui algoritm genetic, acesta trebuie supus unei etape experimentale, în care se va observa influenţa diferiţilor parametrii ai algoritmului asupra rezultatului final, pentru a se ajusta și astfel ajunge la varianta ideală dedicată problemei de rezolvat. Acești factori sunt: rata de încrucişare, rata de mutaţie, strategia de selecţie, dimensiunea populației, rata elitismului și criteriul de oprire. De altfel, algoritmi genetici intră în categoria tehnicilor de inteligenţă artificială care sunt puternic influenţate de problema analizată. Cu toate acestea există o serie de reguli sau recomadări care s-au dovedit de succes pentru o categorie foarte largă de probleme.

Rata de încrucișare

Rata de încrucișare controlează numărul de perechi selectate pentru a fi părinți, și care finalizează prin a produce urmași. Deci, este influențată etapa de explorare a AG. În general, rata de încrucișare variază între 0,7 și 0,95.

Rata de mutație

Un număr prea mare de mutaţii elimină efectul pozitiv al selecţiei după funcţia de adaptare. Impunerea unei valori mari pentru rata de mutaţie are ca efect formarea unei populaţii pe baze dominant stohastice, iar încrucișarea nu reuşeşte să promoveze schemele cele mai adaptate, deoarece aceste scheme sunt distruse de mutaţii. Prea puţine mutaţii limitează gradul de diversificare a populaţiei, care nu mai evoluează. Deci, rata de mutație influențează diversitatea. Valoarea ei uzuală este în intervalul dintre 0,001 și 0,01 pentru codificarea binară și 0,01 și 0,1 pentu codificarea cu valori numerice.

Strategia de selecție

Strategia de selecția afectează diversitatea și convergența, astfel ea este foarte importantă în ceea ce privește eficiența AG. Cele mai uzuale strategi folosite sunt selecția după rang și selecția ruletei.

Mărimea populațiilor

Populaţiile de dimensiuni mici au tendința de a deveni dominate de o soluţie unică şi nu au robustețea necesară pentru adaptare (se aplică aceeaşi regulă din cazul sistemelor naturale, conform căreia nici o specie nu poate supravieţui dacă nu există o populaţie de dimensiuni minimale).

În cazul problemelor de complexitate mărită, se recomandă folosirea unor populaţii cu un număr foarte mare de cromozomi (timpii de calcul cresc, spaţiul de căutare este mult mai bine acoperit).

Există cazuri în care s-au raportat performanţe foarte bune pentru populaţii cu 50 sau 10000 de cromozomi, dimensiunea recomadată pentru populaţia folosită este între 100 şi 200 de cromozomi.

Rata elitismului

La formarea unei noi populaţii se recomadă folosirea elitismului, deoarece ajută la păstrarea celor mai bune soluții, respectiv pierderea lor nevoit. Rata elitismului este de obicei în intervalul 1% și 5% din populație. Adică, maxim 5% din cei mai adaptați indivizi vor trece mereu în următoarea generație.

Criteriul de oprire

Criteriul de oprire influențează convergența și timpul de lucru al algoritmului.

### 12.5. Implementarea algoritmilor genetici

La fel ca în cazul celorlalte tehnci de IA, și AG din punctul de vedere al implementării lor, se împart în două mari categorii – software și hardware.

***Algoritmii genetici software*** sunt programe informatice / aplicații software care rulează pe procesoare de uz general, precum calculatoare personale sau alte dispozitive similare. Această abordare în implementarea AG este cea mai comună, ea fiind folosită atât în cercetare cât și în aplicațiile industriale. Popularitatea ei rezultă din faptul că este flexibilă, ușor de modificat și scalabilă.

Instrumentele informatice și limbajele de programare folosite sunt: Phyton, MATLAB, C/C++, Java, R, și LabVIEW.

Phyton este varianta cea mai folosită atunci când se lucrează cu AG. Aceasta datorită accesibilității și comunității active. În acest limbaj sunt două biblioteci dedicate dezvoltării de AG, și anume DEAP și PyGAD.

DEAP (Distributed Evolutionary Algorithms in Phyton) este cadrul de bază pentru calculul evolutiv. Acesta suportă algoritmi genetici clasici, programarea genetică și AG multi-obiectiv, astfel este ideal pentru dezvoltarea rapidă de prototipuri și cercetare, datorită și faptului că este modular, felxibil și ușor integrat cu NumPy.

PyGAD este o librarie Phyton, ușor de folosit și prietenoasă cu utilizatorii. Caracteristicile de bază ale acesteia sunt: permite dezvoltarea de funcții de fitness personalizate, la fel și pentru operatorii de încrucișare și mutație. Astfel, ea este ideală pentru aplicațiile inginerești și poate fi folosită cu ușurință în procesul de predare. Avantajele acestei bibloteci provind de la faptul că suportă GPU, precum și AG codificați cu valori numerice.

MATLAB-Global Optimization Toolbox este instrumentul de bază a mediului MATLAB, care permite implementarea de înaltă calitate a AG și a altor metaeuristici. Această unealtă informatică permite dezvoltarea de cromozomi binari și cu valori numerice, implementarea de constrângeri, funcții obiectiv și fitness hibride, precum și calculul paralel. În consecință, această unealtă poate fi folosită cu succes în inginerie, predare, dar și optimizarea în sistemele energetice. Avantajele acestei abordări sunt: existența unei interfețe grafice cu utilizatorul ușor de folosit, integrarea cu Simulink și maturizarea aplicației, care o face de încredere pentru uz industrial.

MATLAB-Simulink împreună cu Stateflow oferă posibilitatea dezvoltării și simulării de AG de control și proiectare.

În articolele de cercetare, aceste două unelte informatice, Phyton și MATLAB sunt cel mai des folosite pentru implementarea și validarea AG.

Limbajul de programare C/C++ are două biblioteci dedicate dezvoltării de algoritmi genetici: GALib și EO. GALib este una dintre cele mai vechi librării dedicate AG din C++, astfel ea este eficientă pentru implementări încorporate la nivel scăzut și în timp real. EO (Evolving Objects - Obiecte în Evoluție) este un cadru informatic bazat pe șabloane C++ pentru calculul evolutiv. Acest cadru suportă AG clasici, strategii evolutive, dar și programarea genetică. Acest cadru are două mari avantaje: (1) este foarte rapid, și (2) este bun pentru integrarea AG în aplicații C/C++ mai mari sau simulări hardware-in-the-loop.

Limbajul de programare Java are două unelte informatice folosite cu succes în dezvoltarea de AG, și anume – ECJ (Evolutionary Computation in Java) și Jenetics. ECJ (dezvoltat de Universitatea George Mason) este un cadru software cuprinzător, el suportând algoritmi genetici clasici, programarea genetică și algoritmi de coevoluție. Jenetics este o bibliotecă Java modernă, care are un design simplu, ușor de utilizat, și orientat pe obiecte. Această unealtă este ideală pentru implementarea industrială, întrucât oferă o integrare ușoară în software la nivel de întreprindere.

Limbajul R are un pachet dedicat AG. Acesta cuprinde instrumente de vizualizare pentru evoluția adaptării, și este optim pentru optimizare statistică, probleme cu obiective multiple, data mining.

Mediul de programare grafică LabVIEW poate fi folosit pentru implementări ale AG în timp real și încorporate (de exemplu, în sisteme de control). Unealta dedicată pentru dezvoltarea AG este parte a Modulului de Proiectare și Simulare a Controlului. Avantajele utilizării LabVIEW în dezvoltarea AG sunt: programarea grafică, legături cu NI DAQ și hardware în timp real. In consecință, folosirea acestui mediu de programare este ideală pentru Hardware-in-the-loop (HIL), electronică de putere, automatizare de laborator.

Avantajele utilizării algoritmilor genetici software provin din faptul că aceștia sunt ușor de dezvoltat, depanat și vizualizat. De asemenea, ele sunt potrivite pentru probleme complexe, de nivel înalt și pot încorpora cu ușirință tehnici hibride și adaptive. Un alt avantaj de loc de neglijat este portabilitatea lor între mașini, și costurile de materiale reduse comparativ cu varianta hardware a AG.

Punctele slabe ale AG software sunt: viteza mai mică de lucru comparativ cu sistemele în timp real sau integrate, consumul mare de energie și timpul mai puțin determinist în aplicațiile de control.

***Algoritmii genetici hardware*** sunt implementați pe dispozitive digitale dedicate precum FPGA și ASIC, sau circuite analogice. Deci, un algoritm genetic

hardware este o implementare a unui algoritm genetic care utilizează circuite electronice — fie configurabile (FPGA), fixe (ASIC) sau programate (MCU/DSP). În loc să fie codificate în limbaje de nivel înalt precum Python sau MATLAB, acestea sunt implementate folosind limbaje de descriere hardware (HDL – Hardware Description Language) sau cod încorporat de nivel scăzut. Aceste variante ale AG sunt utilizate în sisteme în timp real, cu latență redusă sau integrate. Cele mai folosite dispozitive sunt FPGA, ASIC, DSP și microcontrolerele.

În implementarea AG hardware fiecare operație a algoritmilor genetici este mapată (asociată) la un modul hardware, și prezentate în tabelul 12.1.

**Tabelul 12.1.** Implementare AG hardware

| **Operația algoritmului genetic** | **Echivalent hardware** |
|---|---|
| Populație | Blocuri sau registre RAM |
| Evaluare fitness | Logică aritmetică dedicată |
| Selecție | Arbori comparatori, LUT-uri |
| Crossover | Registre de deplasare, porți logice |
| Mutație | Logică bit-flip, RNG-uri |
| Înlocuire | Multiplexoare de înlocuire + managementul memoriei |

FPGA-urile (Field-Programmable Gate Array) sunt cea mai comună opțiune pentru dezvoltarea AG hardware. Limbajele HDL folosite sunt VHDL, Verilog sau SystemVerilog. Punctele forte în această abordare sunt evaluarea în paralel a adaptării cromozomilor și a operației de reproducere, respectiv faptul că pot fi reprogramate.

ASIC (Application Circuit Integrat Specific ) sunt dispozitive care sunt caracteristice dacă implementarea AG este permanentă pentru o sarcină specifică. Această aplicație este ultra-rapidă și eficientă energetică. principalul dezavantaj este costul ridicat de proiectare și faptul că nu se poate reconfigura.

Microcontrolerele (MCU) oferă posibilitatea dezvoltării doar de algoritmi genetici simpli, cu populații mici. Acestea sunt programate în C, iar execuția lor se face secvențial. Principalul avantaj este consumul redus de energie, iar functele slabe sunt viteza redusă și paralelismul limitat.

DSP (Procesoare Digitale de Semnal) sunt folosite pentru căutarea evolutivă și prelucrarea semnalelor în timp real. Principalele aplicații sunt în filtrarea adaptivă, reglare audio/video.

Avantajele acestei abordări (AG hardware) sunt rapiditatea datorită execuției paralele, eficiența energetică și previzibilă, ceea ce le fac potrivite pentru control sau optimizare în timp real. Comparativ, dezavantajele AG hardware rezultă din lipsa de flexibilitate (arhitectură fixă, post-fabricare (ASIC)), necesitatea de cunoștințe de HDL (de exemplu, VHDL, Verilog) și complexitatea în proiectare și modificare.

Surse bibliografice cu ajutorul cărora se poate aprofunda subiectul algoritmilor genetici sunt [59] – [67].

### 12.6. Întrebări și exerciții

Întrebări

1. Dimensiunea minimă a unei populații este de 20 de indivizi.
   Adevărat / Fals.

2. Generația unui AG cuprinde patru etape de bază, printre care se numără Selecția / Înlocuirea / Codificarea problemei / Evaluarea / Generarea unei noi populații.

3. Un factor care influențează eficiența unui AG este dimensiunea populației.
   Adevărat / Fals.

4. Cea mai populară abordare în implementarea software a AG este prin implicarea Phyton.
   Adevărat / Fals.

5. Implementarea hardware a AG este mai eficientă decât cea software.
   Adevărat / Fals.

# 13

# Algoritmi genetici
## Aplicații în managementul energiei

Acest capitol prezintă exemple de utilizare a algoritmilor genetici în rezolvarea problemelor din sistemele electroenergetice, respectiv legate de managementul energiei. Prima parte a capitolului enumeră aplicațiile AG în inginerie energetică, următoarele patru subcapitole descrie succint exemple extrase din literatura de specialitate.

### 13.1. Introducere

Algoritmii genetici sunt tehnici de optimizare inspirate de selecția naturală și genetică. Sunt foarte versatili și utilizați în multe domenii în care metodele tradiționale de optimizare au dificultăți, în special în spații de probleme complexe, neliniare și multimodale.

Domeniile mari de utilizare ale AG sunt:

- Optimizare – AG sunt utilizați cel mai cunoscut ca optimizatori metaeuristici pentru rezolvarea problemelor neliniare, de înaltă dimensionalitate, nediferențiabile sau multimodale.
- Control și reglare - AG sunt implicați în reglarea parametrii controlerelor sau ai sistemelor de decizie, în special acolo unde tehnicile clasice de control sunt dificil de aplicat.
- Proiectare și configurare - AG poate căuta modele structurale sau parametrice sub constrângeri multiple.
- Prognoză și estimare – AG ajută la selectarea parametrilor sau intrărilor modelului sau la construirea de modele hibride pentru predicții precise.
- Selecția caracteristicilor și antrenamentul modelului - AG este frecvent utilizat pentru a selecta variabile de intrare importante sau pentru a antrena modele de învățare automată.

- Clasificare și recunoaștere a modelelor - deși mai puțin frecvente decât sarcinile de optimizare, AG sunt utilizați pentru a dezvolta clasificatori bazați pe reguli sau ML.
- Căutare și descoperire – AG acționează ca mecanisme globale de căutare pentru a descoperi noi configurații, secvențe sau soluții în spații complexe.

Utilizarea AG în rezolvarea problemelor din sistemele electrenergetice pe cele patru subramuri: producere, transport, distribuție și utilizare este exemplificată în următoarele paragrafe.

Producerea energiei electrice

- Optimizare
  - Dispecerizare economică – AG minimizează costul combustibilului, respectând în același timp constrângerile de sarcină și operaționale.
  - Angajament unitar - AG stabilește care generatoare să fie pornite/oprite pe un interval de timp.
  - Programarea optimă a generării de energie - pentru unități termice, hidroelectrice și regenerabile.
- Control
  - Control automat al generării - AG reglează parametrii controlerului pentru stabilitatea frecvenței.
- Modelare hibridă
  - AG + ANN/Fuzzy - pentru prognoza pe termen scurt a sarcinii, pentru a îmbunătăți programarea generatorului.

Transportul energiei electrice

- Planificare și optimizare
  - Circulația optimă de putere – rezolvare de problemele neliniare, inclusiv limitele de tensiune, pierderile de linie etc.
  - Planificarea extinderii rețelei de transport – decizii privind locația noi linii de transport în mod rentabil.
  - Managementul congestiei - optimizarea redispecerizării generării sau amplasarea FACTS.
- Control și stabilitate
  - Reglarea stabilizatorului sistemului energetic - utilizarea AG pentru a optimiza parametrii pentru amortizarea oscilațiilor.
  - Coordonarea dispozitivelor FACTS - dimensionarea optimă și setările de control pentru SVC, TCSC, UPFC etc.

Distribuția energiei electrice

- Configurare și optimizare
  - Reconfigurarea rețelei - minimizarea pierderilor și echilibrarea sarcinii prin comutare optimă.
  - Amplasarea și dimensionarea condensatoarelor - pentru a reduce pierderile și a îmbunătăți profilul de tensiune.
- Integrarea energiei regenerabile
  - Amplasarea generării distribuite - optimizarea amplasării și dimensiunii sistemelor fotovoltaice, eoliene sau hibride.
  - Planificarea microrețelelor - optimizarea strategiei de amplasare și control în condiții de limitări tehnice și economice.
- Prognoza sarcinii
  - Predicția cererii de energie electrică pe termen scurt - analiza genetică utilizată pentru selecția caracteristicilor sau modele de prognoză hibridă.

Utilizarea energiei electrice

- Managementul cererii de energiei electrică
  - Programarea consumului de energie electrică - organizarea optimă a consumului pentru reducerea vârfurilor de sarcină sau a costurilor.
  - Controlul inteligent al aparatelor electrocasnice - evoluția regulilor de control pentru dispozitivele inteligente de uz casnic.
- Eficiență energetică
  - Managementul energetic la clădiri - optimizarea programelor de funcționare a sistemelor HVAC, a iluminatului și a aparatelor electrocasnice.
- Managementul vehiculelor electrice
  - Programarea încărcării - optimizarea profilurilor de încărcare a EV pentru a reduce vârfurile de sarcină sau costurile.

Particularizarea aplicațiilor AG pentru managementul energiei electrice înseamnă concentrarea strict pe acțiunile legate de managementul energiei, și anume planificarea, programarea, controlul și optimizarea utilizării energiei și a resurselor. Sinteza acestor aplicații este prezentată în tabelul 13.1.

**Tabelul 13.1.** Aplicații ale AG în managementul energiei

| Sector | Aplicație | Sursa | Rol AG | Tip AG |
|---|---|---|---|---|
| Producerea energiei electrice | Programarea hidrotermală | Sistem energetic Bhakra–Beas, India (aplicație reală) | Optimizarea dispecerizării apei/ combustibilului în condiții de constrângeri | AG multi - obiectiv |
| | Dispecerizare economică pe baza vântului | Red Eléctrica, Spania (aplicație reală) | Dispecerizare cu incertitudine privind regenerabilele | AG stocastic |
| | Programarea unităților de cogenerare | Cercetare: IEEE Transactions on Power Systems | Optimizarea programului de producție termică și electrică | AG cu valori numerice |
| | Angajamentul unității pe baza unor constrângeri privind emisiile | Cercetare: Applied Energy, diverse studii | Minimizarea costurilor și emisiilor în comun | NSGA-II (AG multi-obiectiv) |
| Transportul energiei electrice | Circulația optimă de putere | Sistemul energetic polonez (aplicație reală) | Optimizarea fluxului cu constrângeri neliniare | AG hibrid (AG + Newton-Raphson) |
| | Alocarea dispozitivelor FACTS | IGMC, Iran (lumea reală) | Controlul traseelor fluxului de energie prin TCSC/SVC | AG cu valori numerice |
| | Planificarea extinderii rețelei de transport | Cercetare: Lucrări IEEE PES | Planificarea extinderilor optime din punct de vedere al costurilor ale rețelei | AG cu valori numerice întregi |
| | Cercetare privind managementul congestiei | Cercetare în sistemele de energie electrică | Generarea de redispecerizare pentru a ameliora congestia rețelei | AG bazat pe constrângeri |
| Distribuție energiei electrice | Reconfigurarea sistemului de distribuție | Taiwan Power Co. (lumea reală) | Reducerea pierderilor și îmbunătățirea | AG cu valori numerice întregi |

| | | | | |
|---|---|---|---|---|
| | | | profilului de tensiune | |
| | Plasarea generatoarelor distribuite (PV/BESS) | Cercetare: Conversia și gestionarea energiei | Localizarea și dimensionarea optimă a surselor regenerabile | AG hibrid (AG + circulația puterii) |
| | Microrețea EMS (Energy Management System) | Proiectul Arctic NRC Canada (lumea reală) | Programarea DER, motorinei și stocării pentru a minimiza consumul de combustibil | AG cu integrare algoritmi de învățare automată |
| | Cercetare privind optimizarea Volt-VAR | Research: International Journal of Electrical Power & Energy Systems | Controlul condensatoarelor și regulatoarelor de tensiune | AG bazat pe constrângeri |
| Utilizare energiei electrice | Programarea inteligentă a electrocasnicelor | Fraunhofer, Germania (Lumea reală) | Reducerea cererii de vârf (peak demand) și a facturilor de energie | AG bazat pe reguli |
| | Managementul încărcării vehiculelor electrice | TU Delft + Stedin, Olanda (Lumea reală) | Evitarea supraîncărcărilor și reducerea cererii de vârf | AG cu obiective multiple |
| | Sistem de management al energiei clădirilor (BEMS) | CleanTech One, Singapore (Lumea reală) | Optimizarea HVAC în limitele de confort termic | AG hibrid (AG + Fuzzy) |
| | Optimizarea programului de răspuns la cerere | Cercetare: IEEE Access, studii Smart Grid | Modelarea participării utilizatorilor și a programării consumului | AG cu gestionarea constrângerilor |
| | Prognoza sarcinii rezidențiale | Cercetare: Soft Computing | Selectarea caracteristicilor de intrare pentru modelul de prognoză | AG + RNA (AG hibrid) |

### 13.2. Aplicații în producerea energiei electrice

În sectorul de producere a energiei electrice, o problemă care ține de activitățile de management al energiei este asigurarea condițiilor necesare pentru a genera cantitatea necesară de energie electrică. O astfel de aplicație este prezentată în lucrarea:

[68] Brennenstuhl M, Otto R, Pietruschka D, Schembera B, Eicker U. Optimized Dimensioning and Economic Assessment of Decentralized Hybrid Small Wind and Photovoltaic Power Systems for Residential Buildings. Energies. 2025; 18(7):1811. https://doi.org/10.3390/en18071811.

În cadrul acestui articol, a fost dimensionat un sistem rezidențial de alimentare cu energie, format dintr-o turbină eoliană de mici dimensiuni, un sistem fotovoltaic, o pompă de căldură, un sistem de stocare a energiei cu baterii și un vehicul electric, pentru diferite amplasamente din Germania și Canada, pe baza unor modele de simulare detaliate și a unor algoritmi genetici, pentru a analiza efectul încărcării bidirecționale asupra dimensiunilor optime ale sistemului și a fezabilității economice. Caracteristicile sistemului hibrid de generare atașat clădirii rezidențiale sunt prezentate în fig. 13.1.

| **Building ID** | **12** |
|---|---|
| Heat pump | Waterkotte Modell DS 5023.5Ai, 22.2 kW |
| Thermal buffer storage | DHW 400l; Heating 1000l |
| Battery storage | None |
| Installed PV power | 13.64 kWp |
| PV orientation | 58.4 m² (48 mod.) orientation 180°, tilt 15°; 49.5 m² (40 mod.) orientation 0°, tilt 15° |
| PV manufacturer and model | Solar Frontier Typ SF155-L |
| Residential useable area | 285.13 m² |
| Heating demand | 22,696 kWh |

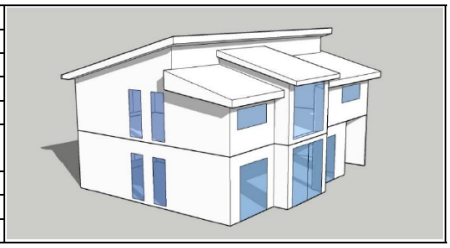


**Fig. 13.1.** caracteristicile clădirii și a sistemului de generare [68]

Algoritmul genetic dezvoltat are rolul de a oferi dimensiunea optimă a elementelor sistemului, astfel codificarea se face cu valori numerice, care variază în intervale date de autori, prin intermediul unor funcții de constrângere. Obiectivul AG este costul sistemului hibrid, deci evaluarea economică este calculată sub forma costului anual echivalent. Acesta constă în cheltuielile anuale pentru investiția, operarea și întreținerea diferitelor componente ale sistemului.

Pentru a identifica parametrii optimi ai AG propus, cu scopul de a găsi echilibrul potrivit între eficiența computațională și apropierea de un optim global, a fost efectuat un studiu iterativ. Acesta a implicat o iterație incrementală în care numărul de generații și dimensiunea populației au fost ajustate în trepte de 20,

variind de la 10 la 110. Simultan, probabilitatea de încrucișare a fost iterată în intervale de 0,1, variind de la pc = 0,6 la pc = 1,0. În plus, probabilitatea de mutație a suferit modificări incrementale în trepte de 0,1, variind de la pm = 0,1 la pm = 0,5. Rezultatele obținute în acest stagiu de evaluare a AG sunt ilustrate în fig. 13.2, unde se observă cum se modifică dimeniunea populație în funcție de numărul de generații, respectiv care este legătura dintre rata de mutație și cea de încrucișare.

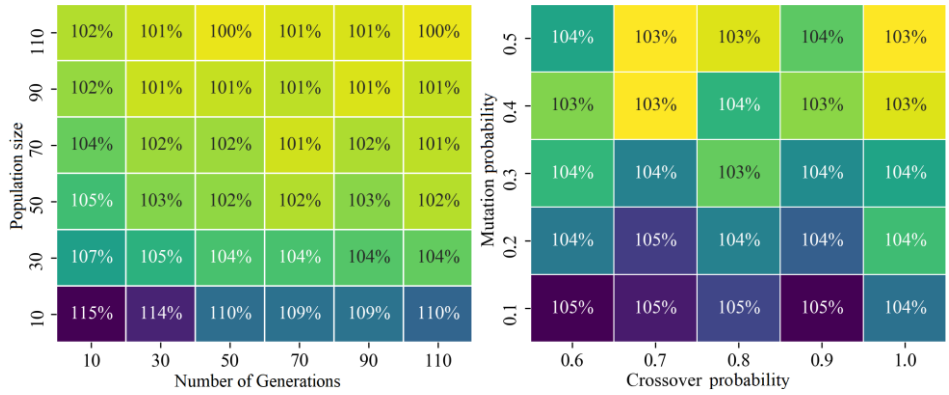



**Fig. 13.2.** Dependența dintre factorii care influențează AG [68]

La finalul acestei etape, AG avea următoarele caracteristici: 90 indivizi într-o populație, 30 de generații, încrucișarea în două puncte, rata de încrucișare 100%, mutația uniformă, rata de mutație 50%, și selecția prin competiție cu mărimea grupului competitor de 5 indivizi. Implementarea AG s-a realizat folosindu-se Phyton.

La sfârșitul cercetării, autorii au furnizat rezultatele, care indică dimensiunea optimă a elementelor sistemului hibrid în funcție de condiții de mediu și de poziționarea clădirii (diferite zone din Germania și Canada).

Un alt exemplu de utilizare a AG în activitățile adiacente producerii energiei electrice este descris în articolul:

[69] Gospodinova E, Nenov D. Forecasting Models and Genetic Algorithms for Researching and Designing Photovoltaic Systems to Deliver Autonomous Power Supply for Residential Consumers. *Applied Sciences*. 2025; 15(9):5033. https://doi.org/10.3390/app15095033.

Scopul cercetării sintetizate în articole este de a lua decizii mai valide și utile prin crearea unui model de prognoză și a unor algoritmi pentru analiza indicatorilor sistemelor PV de mici dimensiuni, ale căror valori actuale sunt prezentate prin serii temporale scurte, și automatizarea proceselor necesare

pentru prognoză și analiză. Astfel, AG utilizat în cercetare este pentru căutarea parametrilor optimi - Valori: FOU (vezi numere fuzzy de tipul 2) a modelui de predicție.

Codificarea soluției s-a realizat folosind valori numerice, astfel: pentru fiecare element al cromozomului, a fost stabilit un interval de variație variabilele considerate (D1—[0, $dl_1$], pentru D2—[0, $dl_2$], pentru n—[2; $n_{max}$], pentru $\alpha_{inferior}$ și $\alpha_{superior}$—[0; 1], pentru k—[2; $k_{max}$], unde $dl_1$ și $dl_2$ sunt numere pozitive egale cu $d_{li} = D_{max} - D_{min}$, i = 1, 2, $n_{max}$ este un număr natural $n_{max} < m - 1$, m este un număr care ia în considerare timpul, iar $k_{max}$ este un număr natural $k_{max} < m - 1$). Evaluarea cromozomilor s-a realizat prin implicarea unei funcții care depinde de valoarea oferită de modelul de predicție. În consecință, soluția optimă este caracterizată printr-o valoare minimă, deci în problemă se urmărește minimizarea funcției obiectiv. Rezultă că, cromozomul care furnizează valoarea minimă a funcției de fitness este recunoscut ca fiind cel mai bun din generația actuală a AG. Pentru selecția părinților s-a utilizat principiul selecției probabilistice: cu cât valoarea funcției de fitness a cromozomului (l = (1, p) este mai mică în populația actuală, cu atât este mai mare probabilitatea ca cromozomul să acționeze ca un cromozom parental în formarea cromozomilor urmași pentru o nouă populație.

La efectuarea operației de încrucișare cromozomul părinte a fost selectat conform următorului algoritm: (1) Sortarea după probabilitatea crescătoare p1 (l = 1, p); (2) Generarea unui număr aleatoriu (uniform distribuit) z din intervalul [0, 1]: z = random([1, 0]); (3) Selectarea cromozomului l ca și cromozom părinte dacă numărul aleatoriu z se încadrează în intervalul l [pl−1; p1 (l = 0, p); p0 = 0]. Pentru a determina cei doi cromozomi părinte, pașii 2 și 3 se repetă de două ori. Pentru operația de încrucișare, coeficientul de încrucișare Rc este fix.

Un număr aleatoriu zc random([1, 0]), c = este generat atunci când se încearcă încrucișarea a doi cromozomi părinți. Dacă Rc > zc, atunci punctul de intersecție este ales c rm (rm ∈ {1, 2, 3, 4, 5) și intersecția se realizează cu doi cromozomi parentali și formarea a doi cromozomi urmași. Pentru operația de mutație, coeficientul de încrucișare Rm este fix. Obținerea de copii se realizează prin încrucișarea într-un singur punct a cromozomilor parentali și nu mai mult de un element (genă) din cromozomul parental suferă mutație.

Diagrama funcțională a AG propus este ilustrată în fig. 13.3. Primul pas este crearea primei populații de cromozomi viabili, urmat de evaluarea cromozomilor, slecție, reproducere, mutație, și desigur verificarea criteriului de oprire, care este număr maxim de generații. La final, AG va oferii parametrii optimi ai numerelor fuzzy folosite în modelele de predicție a sistemelor PV.

Eficiența soluției date de AG a fost verificată prin implementarea unui sistem PV în MATLAB, și estimarea producției prin utilizarea modelelor de predicție cu parametrii dați de AG.

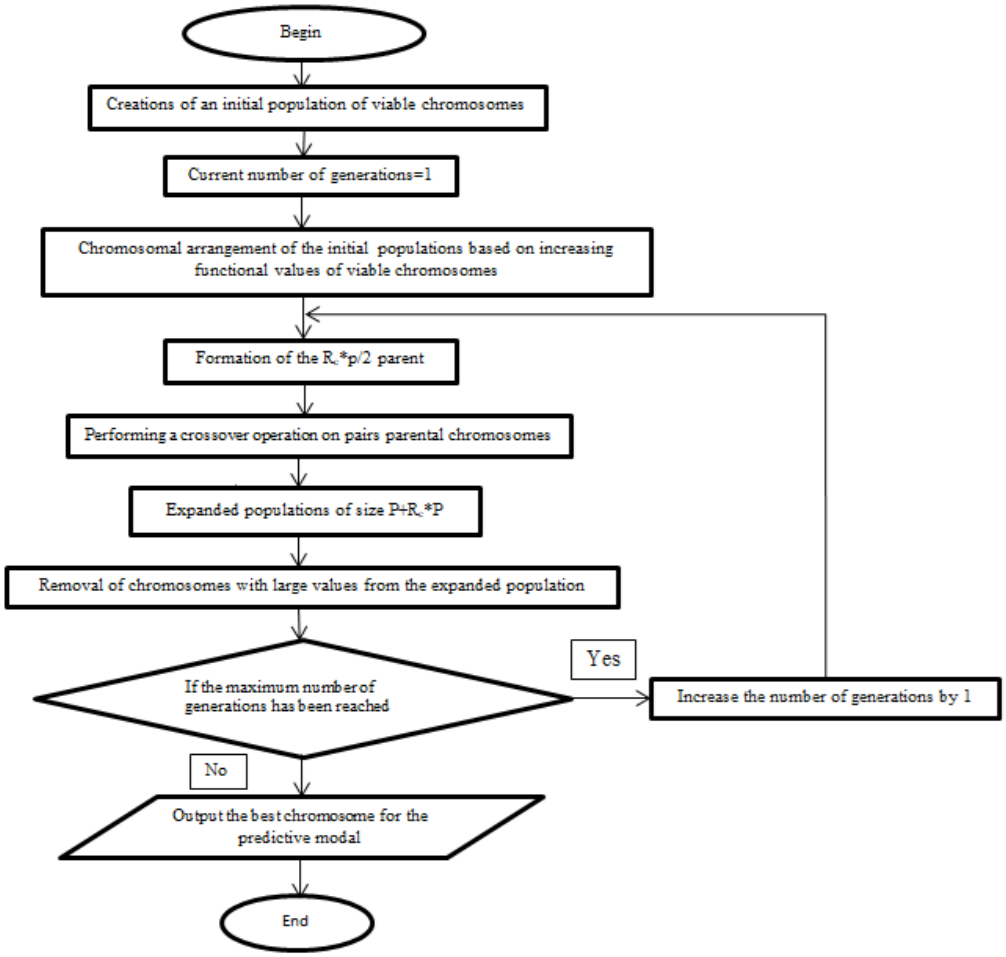



**Fig. 13.3.** Diagrama funcțională a algoritmului genetic propus [69]

La finalul articolului, autorii au concluzionat că algoritmi dezvoltați le-au permis să analizeze comportamentul grupurilor de indicatori pentru evaluarea eficienței energetice a sistemelor fotovoltaice mici pentru consumatorii casnici.

O altă aplicație a AG prezentată în literature de specialitate este descrisă în articolul

[70] Silva CH, Mendes SFS, Garcés Negrete LP, López-Lezama JM, Muñoz-Galeano N. Optimizing Photovoltaic Generation Placement and Sizing

Using Evolutionary Strategies Under Spatial Constraints. *Energies*. 2025; 18(8):2091. https://doi.org/10.3390/en18082091.

Acest studiu prezintă o metodologie pentru optimizarea amplasării și dimensionării sistemelor fotovoltaice în rețelele de distribuție a energiei electrice. Pe lângă constrângerile tehnice și bugetare, abordarea propusă încorporează restricții spațiale georeferențiate pentru a determina amplasarea și capacitatea optimă a unităților de generare.

Funcția de optim a AG se referă la reducerea pierderilor de energie, care va rezulta în reducerea costurilor. În plus, s au fost adăugate multe constrîngeri tehnici și geospațiale, prin care să se limiteze și dirijeze procesul de explorare și exploatare a AG. Rolul AG este de a ogeri poziția și puterea oprimă a sistemului fotovoltaic, el se folosește codificarea binară, fig. 13.4. ilustrează operatorul mutație folosit în studiu.

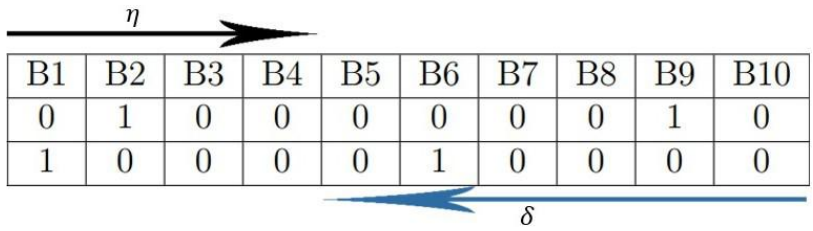



**Fig. 13.4.** Operatorul mutație [70]

În studiul prezentat în articolul menționat a fost folosit un AG elitist, algoritmul în pseudocod a acestuia este ilustrat în fig. 13.5.

**Algorithm 3** The Genetic Algorithm with Elitism

```
μ ← number of selected parents
λ ← number of offspring generated by the parents
P ← {}
for μ + λ times do
    P ← P ∪ new random individual
end for
Best ← Arbitrary value
while stopping condition not satisfied do
    for each individual P_i ∈ P do
        Evaluate(P_i)
        if Quality(P_i) > Quality(Best) then
            Best ← P_i
        end if
    end for
    Q ← the μ fittest individuals in P
    for (μ + λ − μ)/2 times do
        P_a ← Selection(P)
        P_b ← Selection(P)
        C_a, C_b ← Crossover(Copy(P_a), Copy(P_b))
        Q ← Q ∪ {Mutation(C_a), Mutation(C_b)}
    end for
    P ← Q
end while
return Best
```

**Fig. 13.5.** Etapele algoritmului genetic propus [70]

Pentru implementarea software au fost folosite uneltele informatice OpenDSS și QGIS.

Metodele prezentate în articol au fost testate pe o rețea standard recunoscută în mediu academic, care este prezentată în fig. 13.6, și care a fost introdusă într-o arie reală, ilustrată în fig. 13.7. Rezultatele studiului (care sunt într-un număr mare) evidențează eficiența metodei propuse în articol.

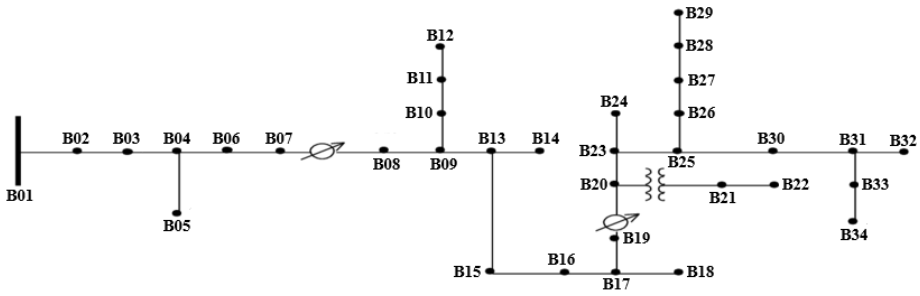



**Fig. 13.6.** Structura sistemului electrenergetic de test [70]

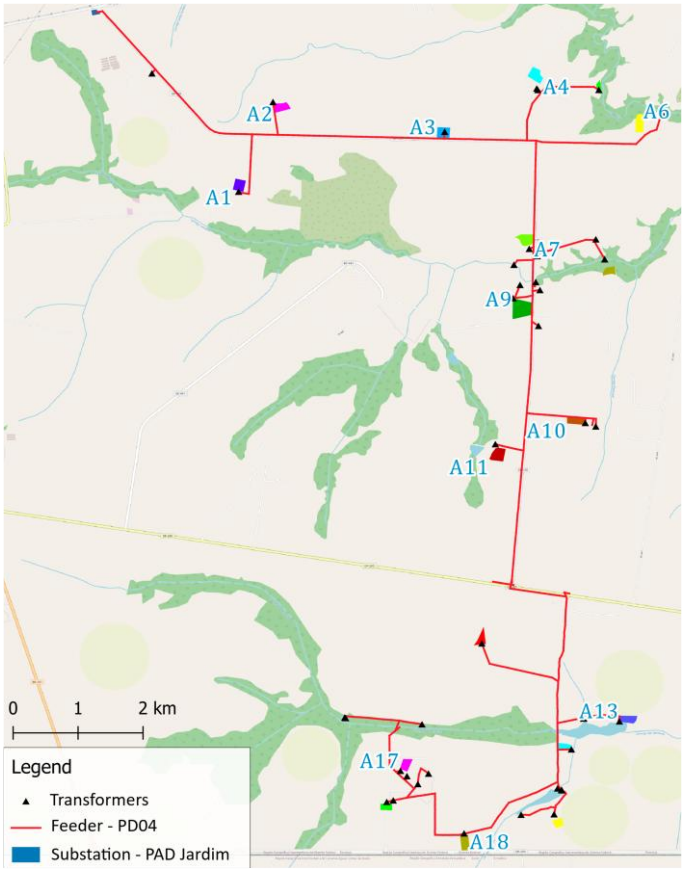



**Fig. 13.7.** Aria reală folosită pentru test [70]

Un alt exemplu de utilizare a AG în dimensionare este prezentat în articolul

[71] Ji C, Nie X, Chen S, Cheng M, Dai Z. Optimization of a Nuclear–CSP Hybrid Energy System Through Multi-Objective Evolutionary Algorithms. Energies. 2025; 18(9):2189. https://doi.org/10.3390/en18092189.

Acest studiu propune un sistem energetic hibrid nuclear-solar (NSHES), care integrează un reactor modular mic cu sare topită de toriu (smTMSR), energie solară cu concentrare (CSP) și stocare a energiei termice (TES). Sunt proiectate și analizate două moduri de funcționare: energie nucleară constantă (modul 1) și energie nucleară ajustată (modul 2). Algoritmul genetic de sortare nedominată II (NSGA-II) este aplicat pentru a minimiza atât probabilitatea deficitului de alimentare cu energie (DPSP), cât și costul nivelat al energiei (LCOE).

Structura sistemului energetic hibrid este prezentată în fig. 13.8., unde se observă toate componentele sistemului și relația dintre ele.

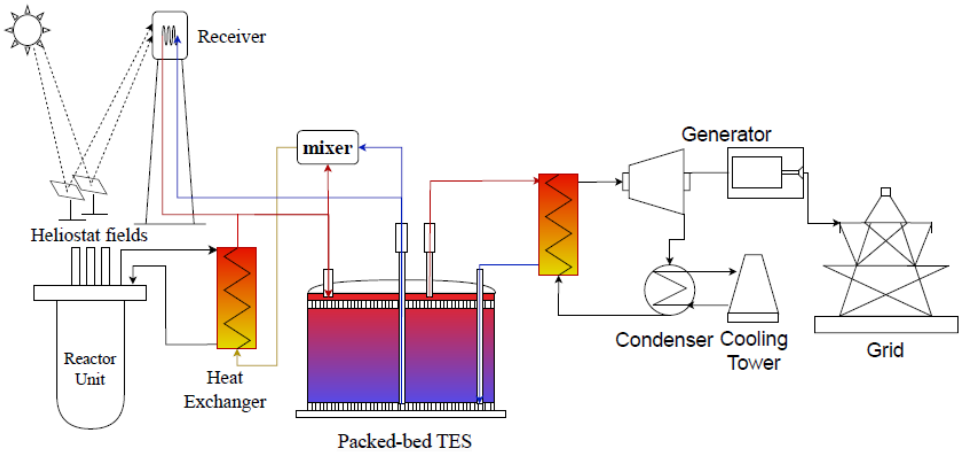



**Fig. 13.7.** Structura sistemului hibrid propus [71]

Baza matematică folosită în studiu pentru dimensionarea sistemului hibrid de generare este prezentată în detaliu în articolul menționat anterior. Diagrama computațională a algoritmului de calcul este prezentată în fig. 13.9. Se observă în diagramă, pași de calcul și relația dintre etapele de calcul a energiei termice, respectiv electrice.

Algoritmul de optimizare abordat în studiu este descris în fig. 13.10, unde apar patru etape principale: optimizarea multi-obiectiv, evaluarea Pareto, determinarea influențelor și clasificarea soluțiilor pentru a găsi optimul global. Optimizarea multi-obiectiv folosește un AG NGSA-II cu două obiective (menionate anterior) și trei constrângeri.

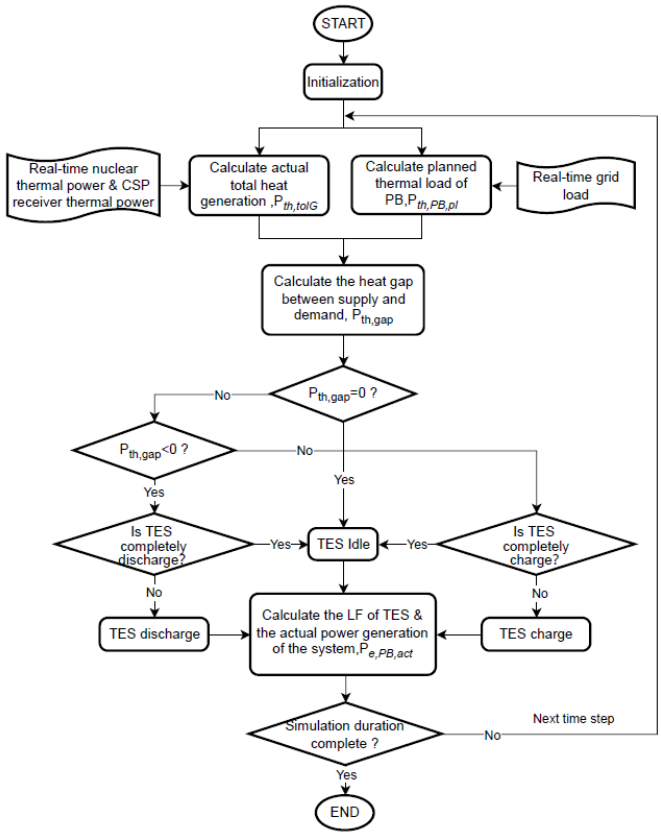



**Fig. 13.9.** Diagrama computațională a sistemului energetic propus [71]

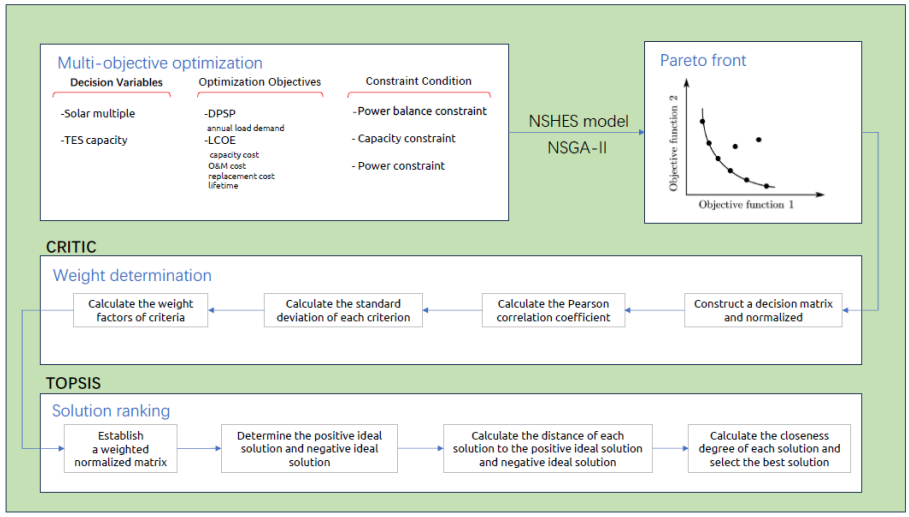



**Fig. 13.10.** Procesul de optimizare abordat în studiu [71]

Rezultatele studiului au indicat fezabilitatea și utilizatea AG propus în dimensionarea sistemului hibrid.

### 13.3. Aplicații în transportul energiei electrice

În transportul energiei electrice, reglarea frecvenței rețelei este o problemă importantă, care poate fi rezolvată și prin implicarea AG, așa cum demonstrează articolul:

[72] Shi T, Wang C, Chen Z. The Multi-Point Cooperative Control Strategy for Electrode Boilers Supporting Grid Frequency Regulation. Processes. 2025; 13(3):785. https://doi.org/10.3390/pr13030785.

Această lucrare propune un model decizional de control al optimizării colaborative multi-obiectiv pentru cazanele cu electrozi, pentru a ajuta la reglarea frecvenței rețelei. Modelul nu numai că îndeplinește cerințele de reglare a frecvenței, dar ia în considerare și constrângeri suplimentare, inclusiv eficiența operațională a cazanelor cu electrozi, beneficiile economice și degradarea echipamentelor. Un algoritm genetic este utilizat pentru a rezolva modelul, iar analiza prin simulare este efectuată folosind sistemul IEEE cu 14 noduri.

Arhitectura strategiei de control a electrozilor boilerelor este ilustrată în fig. 13.11, în care se observă algoritmul de optimizare propus de autori, și modelarea matematică a tuturor mărimilor fizice folosite în studiu.

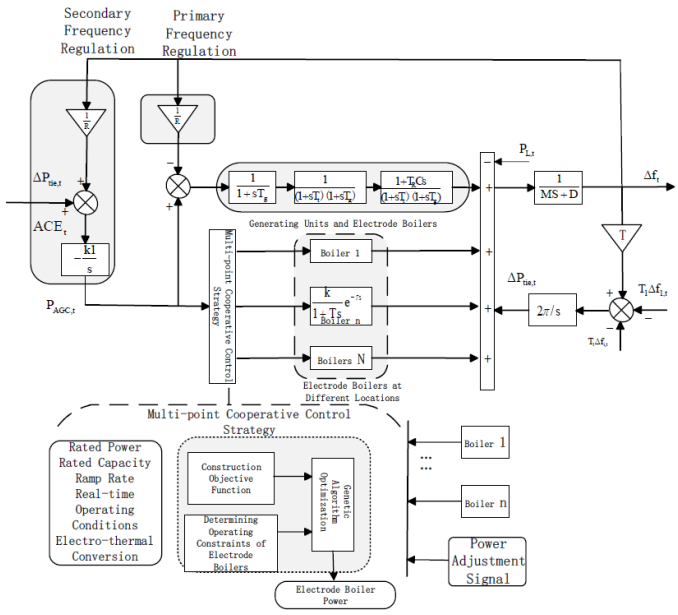



**Fig. 13.11.** Arhitectura sistemului de control a electrozilor [72]

În studiu se folosește un scenariu de reglare a frecvenței, care implică cazane cu cinci electrozi și rezervoare de stocare termică, necesitând o alocare coordonată a puterii pentru mai multe dispozitive sub constrângeri dinamice de nivel doi. Prin proiectarea unei strategii de codificare adecvate a AG bazată pe inițializarea soluției fezabile și încorporarea funcțiilor de penalizare dinamice pentru a gestiona constrângerile de rată de rampă, GA poate genera soluții fezabile în 30 de secunde. În plus, AG permite extinderea flexibilă a funcției obiectiv, punând bazele optimizării multi-obiectiv. Diagrama de funcționare a AG este prezentată în fig. 13.13. Codificarea folosită în algoritm este prin utilizarea de valori numerice reale. Operatorul de selecție este după rang și ruleta, iar mutația este prin funcția Gauss.

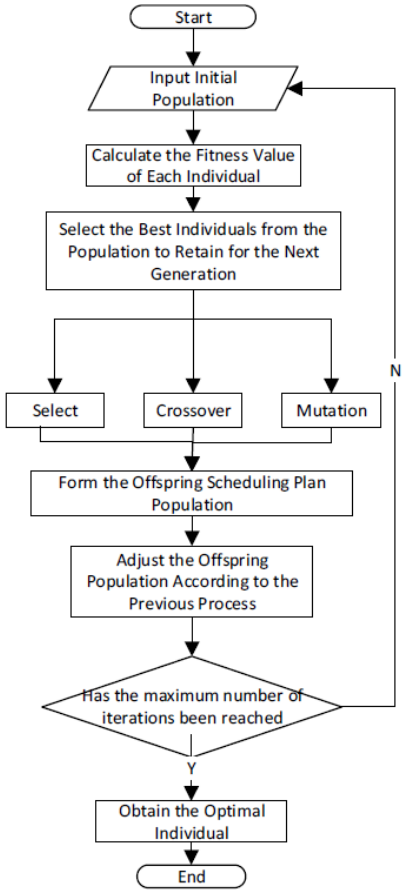



**Fig. 13.12.** Diagrama operațională a algoritmului genetic propus [72]

Algoritmul a fost testat și eficientizat pentru a oferi varianta cea mai bună. Într-adevăr, a fost variat numărul de indivizi între trei valori: 10, 30 și 50, la final alegându-se ca populațiile să aibă câte 30 de indivizi.

Structura sistemului electroenergetic considerat este cel din fig. 13.13. rezultatele simulării a arătat că alocarea optimizată cu AG au fost superioare rezultatelor obținute prin alocarea clasică.

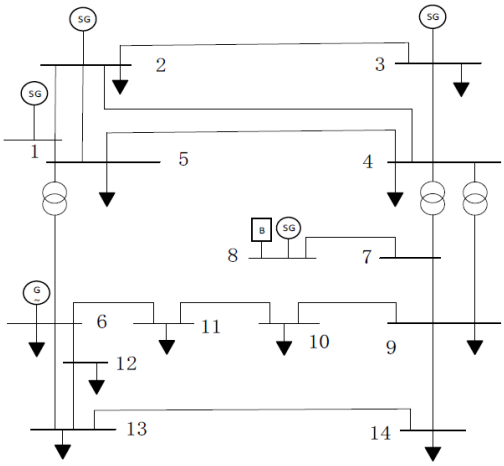



**Fig. 13.13.** Topologia sistemului energetic test cu 14 noduri [72]

În sistemele de transport al energiei electrice stabilitatea este un deziderat, care se poate realiza prin utilizarea compensatoare statice sincrone. Un studiu care folosește această este descris în articolul:

[73] Urrea-Aguirre C, Saldarriaga-Zuluaga SD, Bustamante-Mesa S, López-Lezama JM, Muñoz-Galeano N. Optimal Placement and Sizing of Modular Series Static Synchronous Compensators (M-SSSCs) for Enhanced Transmission Line Loadability, Loss Reduction, and Stability Improvement. Processes. 2025; 13(1):34. https://doi.org/10.3390/pr13010034.

Această lucrare abordează amplasarea și dimensionarea optimă a Compensatoarelor Serie Sincrone Statice Modulare (M-SSSC) pentru a îmbunătăți performanța sistemului energetic. Metodologia propusă optimizează patru obiective cheie: reducerea capacității de încărcare a liniilor de transmisie, minimizarea pierderilor de putere, atenuarea abaterilor de tensiune și îmbunătățirea stabilității tensiunii utilizând indicele L. Metodologia este validată pe două sisteme: rețeaua de testare IEEE cu 14 magistrale și o subzonă a rețelei

electrice columbiene, caracterizată de infrastructură îmbătrânită și provocări operaționale. Pentru aceasta s-a folosit un AG multi-obiectiv.

Structura de bază a unui M-SSSC este descrisă în fig. 13.14.

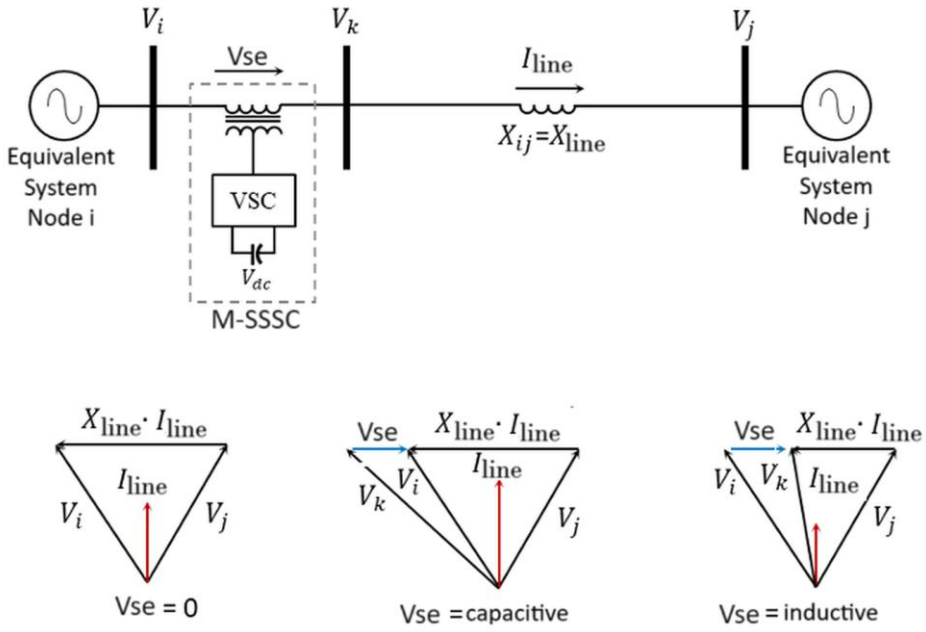



**Fig. 13.14.** Diagrama operațională a M-SSSC utilizat în studiu [73]

Algoritmul genetic folosit are structura clasică a unui algoritm multi-obiectiv, cu populație de dimensiune fixă, selecție după rang, încrucișarea într-un singur punct a cărui poziție se schimbă aleator. Algoritmul este caracterizat și prin șase funcții de contrângere prin care valorile variabilelor sunt menținute în intervale bine definite.

În studiu, pentru a demonstra aplicabilitatea abordării propuse, au fost efectuate mai multe teste în două sisteme de transmisie: sistemul de testare pe magistrală IEEE-14 și o subzonă a sistemului de transmisie columbian cunoscută sub numele de GCM (Guajira–Cesar–Magdalena). Inițial, doar o funcție obiectiv a fost considerată a dispozitivului M-SSSC pentru a reduce încărcarea liniilor de transmisie; ulterior, sunt introduși și analizați și ceilalți termeni ai funcției obiectiv. Varianta finală a AG este caracterizată prin următoarele: o populație de 10 indivizi, 20 de generații, rata de încrucișare de 0,7, un factor de distanță suplimentar pentru o încrucișare de 0,4, un procent de mutație de 0,3 și o rată de mutație de 0,1.

Pentru testare, metoda dezvoltată a fost implementată în MATLAB, rezultatele simulării subliniind superioritatea AG.

### 13.4. Aplicații în distribuția energiei electrice

Stabilitatea sistemului electroenergetic poate fi menținută prin diferite acțiuni de coordonare în toate sectoarele, inclusiv în cel de distribuție a energiei electrice. O aplicație a AG care presupune implicarea stațiilor de încărcare a vehiculelor electrice pentru a menține stabilitatea sistemelor de distribuție, este prezentată în articolul următor.

[74] Zhan J, Huang M, Sun X, Chen Z, Zhang Z, Li Y, Zhang Y, Ai Q. Coordinated Interaction Strategy of User-Side EV Charging Piles for Distribution Network Power Stability. *Energies*. 2025; 18(8):1944. https://doi.org/10.3390/en18081944.

Acest articol propune o strategie de reglare a încărcării și descărcării vehiculelor electrice, care ia în considerare interacțiunile dintre vehicul și rețeaua rutieră și predicția spațiotemporală a cererii de încărcare. Prima etapă a studiului a fost stabilirea separat a unor modele dinamice ale rețelei rutiere și rețelei electrice, folosindu-se stațiile de încărcare ca legătură pentru a realiza relația de cuplare dintre rețeaua rutieră și rețeaua electrică. Apoi, pe baza modelului dinamic al rețelei rutiere și a caracteristicilor de deplasare ale locuitorilor urbani, sunt prezise caracteristicile de distribuție spațiotemporală a încărcării vehiculelor electrice. Următorul pas a studiului a fost dezvoltarea unui model de reglare multi-obiectiv pentru optimizarea încărcării și descărcării, cu veniturile stațiilor de încărcare, costurile de încărcare ale utilizatorilor și fluctuațiile de tensiune ca funcții obiectiv. În cele din urmă, metoda a fost aplicată în regiuni reale pentru verificare prin simulare folosind algoritmul genetic de sortare nedominată II (NSGA-II) bazat pe puncte de referință.

Algoritmul multi-obiectiv urmărește minimizarea costurilor de încărcare, maximizarea veniturilor din stații și reglarea fluctuațiilor de tensiune din rețeaua de distribuție. Restricțiile problemei țin de variația puterilor activă și reactivă, respectiv a tensiunii. Diagrama funcțională a strategiei de optimizare este ilustrată în fig. 13.15, unde se poate observa câteva aspecte ale metodologie: mereu este verificată starea de încărcare, SOC, a vehiculelor electrice, la final se verifică condiția Pareto, iar soluția este aleasă prin metoda Țintei gri (Grey-Taget method). Algoritmul a fost testat folosindu-se populații de 100 de indivizi, și 200

de generații, rata de împerechere de 0,5, rata de mutație de 0,02, iar procentajul de mutație de 0,2 (20% din genele unui cromozom pot suferi mutații).

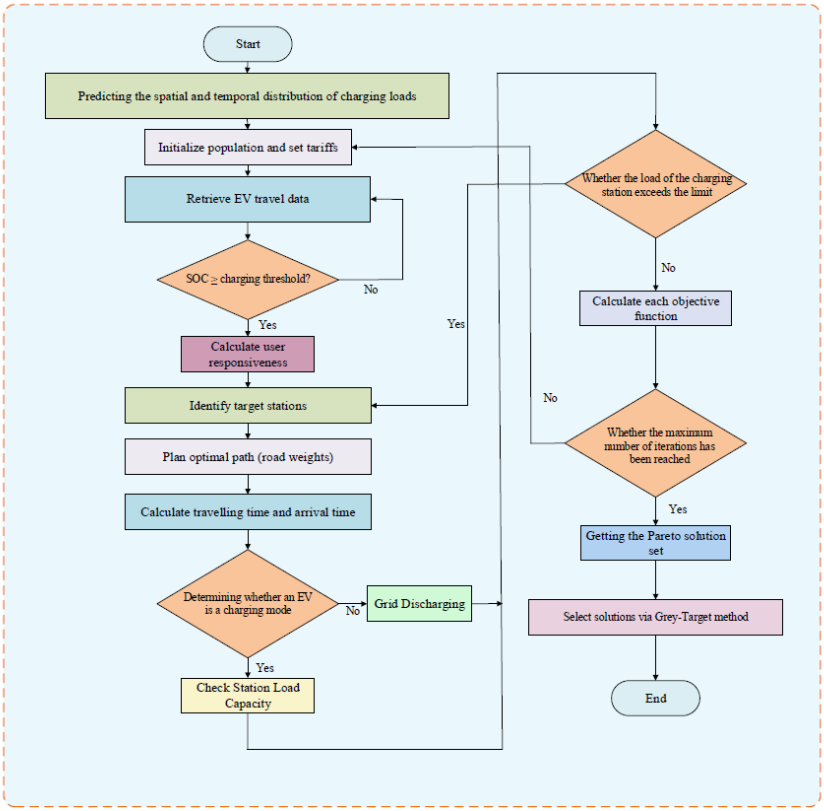



**Fig. 13.15.** Diagrama funcțională a strategiei de optimizare [74]

Algoritmul a fost implementat și testat, iar apoi rezultatele comparate cu alte metode de optimizare. Analiza rezultatelor a indicat superioritatea metodei propuse.

Un alt exemplu de utilizare a AG este planificarea optimă a rețelelor active de distribuție prin folosirea sistemelor hibride de stocare a energiei propus în articolul:

[75] Miao L, Di L, Zhao J, Liu H, Hu Y, Wei X. Optimal Scheduling of Active Distribution Networks with Hybrid Energy Storage Systems Under Real Road Network Topology. *Processes*. 2025; 13(5):1492. https://doi.org/10.3390/pr13051492.

Articolul menționat prezintă o metodologie de planificare prin implicarea unui algoritm genetic de sortare nedominată III (NSGA-III), care este utilizat

pentru a rezolva modelul hibrid de planificare a optimizării sistemului de stocare. Astfel, cadrul optim de programare pentru sistemul hibrid de stocare este un model de optimizare multi-obiectiv, în care variabilele de decizie sunt nodurile de încărcare și descărcare, timpul și puterea corespunzătoare a sistemului de stocare, nodurile de încărcare și descărcare, precum și puterea sistemului de stocare și a rețelei de alimentare cu energie electrică. Mai mult, funcția obiectiv ia în considerare beneficiul net al programării și abaterea totală de tensiune.

Diagrama funcțională a metodologie propuse este prezentată în fig. 13.16.

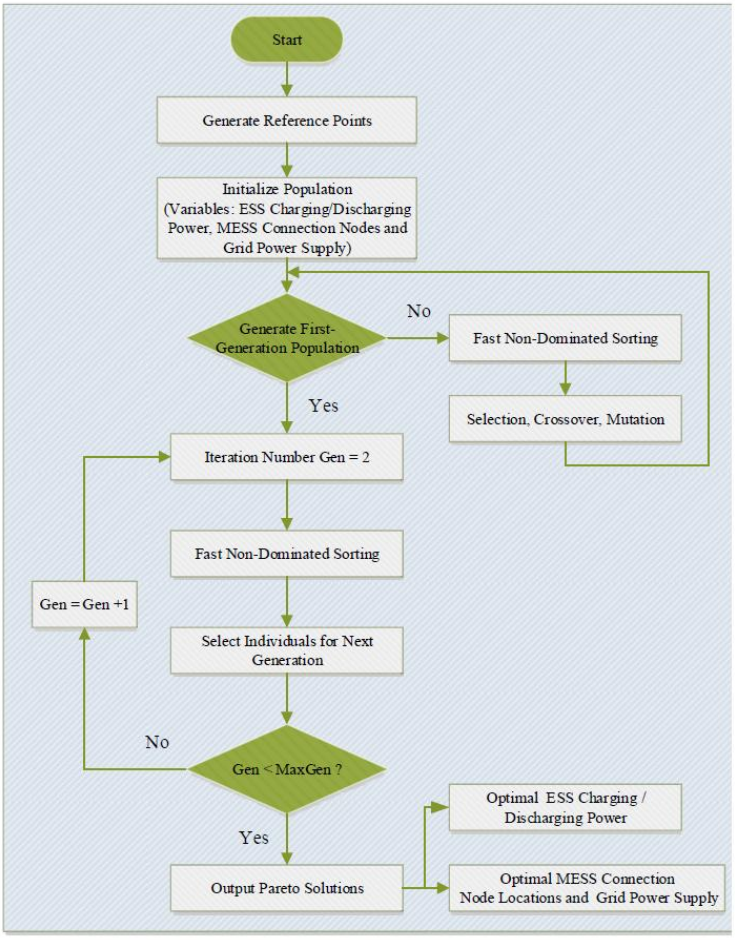



**Fig. 13.16.** Diagrama funcțională a strategiei de optimizare [75]

Analiza diagramei funcționale ne arată că criteriul de oprire este numărul maxim de generații, iar la final prin considerea soluției Pareto, se alege acea variantă care oferă încărcarea/descărcarea optimă a sistemului de stocare și

conectarea optimă cu rețeaua de distribuție. Precum la toate studiile anterioare, autorii demonstrează superioritatea metodei propuse de ei.

### 13.5. Aplicații în utilizarea energiei electrice

Un obiectiv mondial este reducerea emisiilor de carbon, aspect care are legătura cu sistemele de utilizare a energiei electrice. Sistemele integrate de energie sunt sisteme energetice în care există o legătură strânsă între producere și utilizare, astfel putând-se controla consumul de resurse și reducerea emisiilor de carbon.

Un studiu în care se discută și prezintă problema emisiilor de carbon și sistemele integrate de energie este descris în lucrarea următoare:

[76] Duan Y, Gao C, Xu Z, Ren S, Wu D. Multi-Objective Optimization for the Low-Carbon Operation of Integrated Energy Systems Based on an Improved Genetic Algorithm. Energies. 2025; 18(9):2283. https://doi.org/10.3390/en18092283.

Această lucrare propune un algoritm genetic îmbunătățit, conceput pentru a optimiza operațiunile cu emisii reduse de carbon cu obiective multiple ale sistemelor integrate de energie, cu scopul de a minimiza atât costurile de operare, cât și emisiile de carbon. Într-adevăr, prin îmbunătățirea mecanismelor de încrucișare, mutație și selecție a urmașilor din algoritmul genetic, capacitatea de optimizare a algoritmului și caracterul rezonabil al soluției sunt semnificativ îmbunătățite, asigurându-se în același timp că constrângerile de egalitate nu produc abateri excesive. Această metodă nu numai că garantează fezabilitatea rezultatelor programării, dar realizează și o operațiune economică, rezolvând eficient problema de optimizare a programării cu emisii reduse de carbon pentru sistemele energetice integrate (fig. 13.17.), cu o zi înainte.

Diagrama funcțională a AG îmbunătățit este ilustrată în fig. 13.18, iar caracteristicile de bază ale acestuia sunt clar descrise în articol.

Codificarea soluției s-a făcut prin valori numerice, alelele fiind valori reale din interiorul intervalelor de variație a mărimilor sistemului integrat. Selecția indivizilor se bazează pe următoarele trei principii: (1) se utilizează algoritmul de sortare rapidă nedominată pentru a sorta indivizii care respectă restricțiile. (2) indivizii care nu le încalcă sunt considerați câștigători. (3) dintre indivizii care încalcă constrângerile, indivizii cu grade mai mici de încălcare a constrângerilor sunt considerați câștigători. Se formează a mulțime cu cromozomi părinți, care

sunt selectați prin selecția prin competiție, cu 2 indivizi. Pentru obținerea de cromozomi urmași, se folosește încrucișarea ciclică.

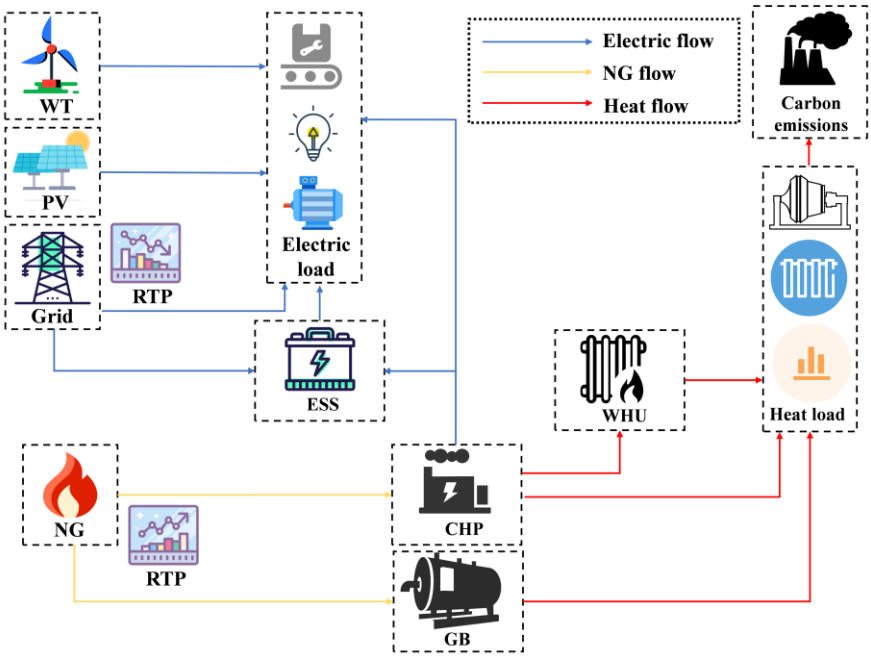



**Fig. 13.17.** Structura de bază a unui sistem energetic integrat [76]

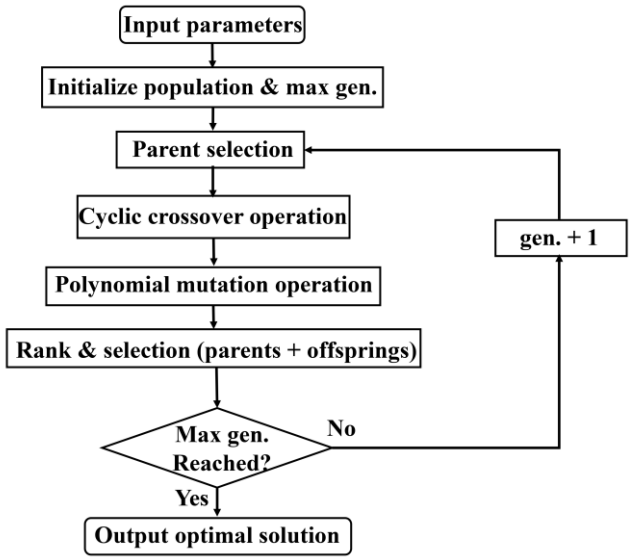



**Fig. 13.18.** Diagrama funcțională a AG îmbunătățit [76]

Operatorul genetic de mutație este mutația polinomială, care este caracteristică codificării cu valori reale.

Întrucât algoritmul de optimizare multi-obiectiv produce un set de soluții nedominate distribuite pe frontul Pareto, este necesară o selecție suplimentară, care utilizează o metodă bazată pe ponderi pentru a selecta soluțiile Pareto.

Algoritmul dezvoltat a fost implementat și testat, cadrul de lucru a acestui proces este descris succint în fig. 13.19 preluată din articol. Precum se observă, au fost folosite mai multe scenarii, prin care s-a demonstrat faptul că metoda propusă este eficace și eficientă.

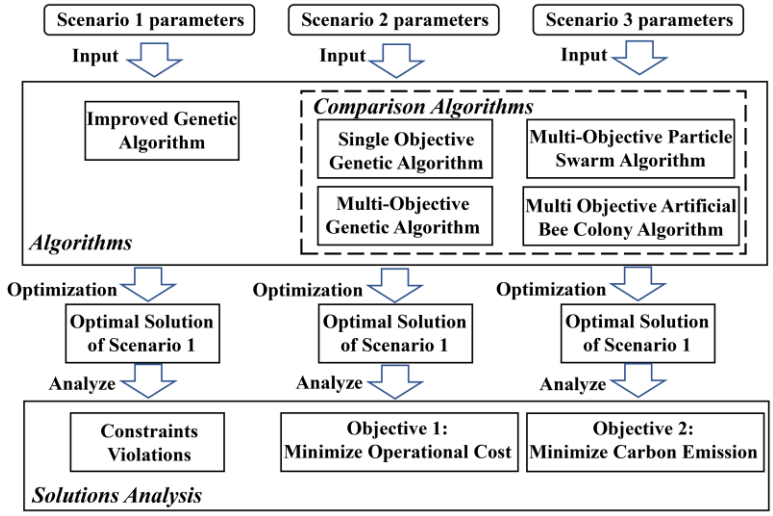



**Fig. 13.19** Cadrul de lucru în implementarea AG propus [76]

Configurația optimă a capacității de stocare a energiei bazată pe algoritmul NSGA-II și pe modurile de operare pentru stocarea electrochimică a energiei, este titlul articolului, care prezintă rezultatele unei cercetări a cărei obiectiv a fost dezvoltarea unei strategii de optimizare pentru dimensionarea sistemelor electrochimice de stocare a energiei.

[77] Hu G, Zhao X, Cheng S, Zhang Q, Zhang Q, Bai J, Shi L. The Optimal Configuration of Energy Storage Capacity Based on the NSGA-II Algorithm and Electrochemical Energy Storage Operational Modes. *Processes*. 2025; 13(5):1432. https://doi.org/10.3390/pr13051432.

Într-adevăr, acest studiu cercetează optimizarea alocării capacității de stocare electrochimică a energiei și stabilește modelul de alocare a optimizării capacității luând în considerare modul de funcționare a stocării electrochimice.

Urmărind un beneficiu net maxim și o fluctuație minimă în timpul conectării la rețea, modelul ia în considerare constrângerile capacității de stocare a energiei și limitele superioare și inferioare ale puterii, constrângerile puterii de încărcare și descărcare și constrângerile stării de încărcare și adoptă un algoritm de tipul NSGA-II pentru a o rezolva. Fluxul computațional al metodei propuse este descris în fig. 13.20.

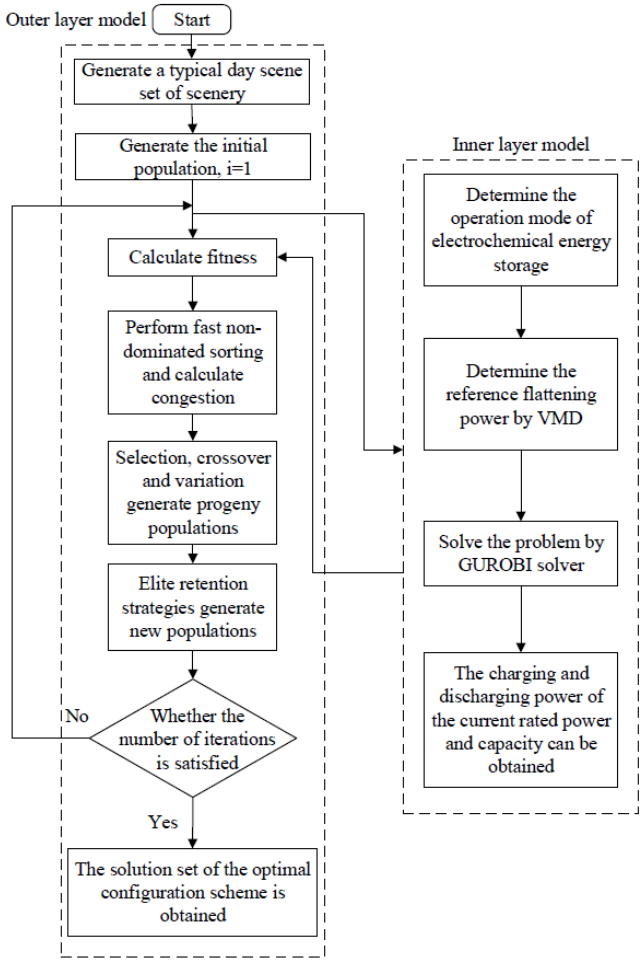



**Fig. 13.20.** Diagrama funcțională a AG propus [77]

Algoritmul genetic dezvoltat este unul elitist, folosind metoda Pareto pentru a găsi soluția optimă. Mai mult, algoritmul are două straturi, cel interior fiind rezolvat prin algoritmul rezolutiv GUROBI 2.0.

La finalul articolului se concluzionează următoarele: stocarea electrochimică a energiei poate netezi eficient fluctuația puterii, și

## 13.6. Întrebări și exerciții

Întrebări

1. Cea mai comună aplicație a AG este în controlul sistemelor energetice.
   Adevărat / Fals.

2. În producerea energiei electrice, AG sunt folosiți în dimensionarea elementelor sistemelor de producere.
   Adevărat / Fals.

3. Problema stabilității sistemului electroenergetic este o problemă care ține de sectorul de transport a energiei electrice.
   Adevărat / Fals.

4. Programarea optimă a alimentării vehiculelor electrice se poate realiza prin implicarea unui AG.
   Adevărat / Fals.

5. Dimensionarea sistemelor integrate de energie se poate soluționa prin utilizarea unui algoritm genetic multi-obiectiv.
   Adevărat / Fals.

# 14

# Tehnici hibride de IA

## Sisteme hibride de IA dedicate managementului energiei

Acest capitol descrie aplicațiile de IA hibride rezultate prin combinarea celorlalte tehnici de IA prezentate în capitolele anterioare. Astfel, se va trata problema asocierii logicii fuzzy cu calculul neuronal, respectiv cu cel genetic, și combinarea dintre rețelele neuronale și algoritmii genetici. La finalul acestui capitol se trag concluzii cu privire la cele prezentate în acest suport de curs.

### 14.1. Introducere

Logica fuzzy este o abordare matematică care modelează raționamentul cu informații vagi sau imprecise, imitând procesul decizional uman. Spre deosebire de logica clasică care utilizează valori binare (adevărat sau fals), logica fuzzy permite valori reale în intervalul [0, 1], permițându-i să gestioneze eficient incertitudinea și adevărurile parțiale. Această tehnică de IA este frecvent utilizată în sistemele de control, probleme de clasificare și luarea deciziilor unde datele sunt imprecise. În ciuda flexibilității sale, utilizarea exclusivă a logicii fuzzy în sistemele fuzzy clasice prezintă mai multe dezavantaje semnificative:

- Numărul mare de reguli - Pe măsură ce numărul de intrări crește, numărul de reguli crește exponențial. Acest lucru face ca sistemul să fie dificil de gestionat, proiectat și întreținut.
- Lipsa capacității de învățare - Sistemele fuzzy clasice nu învață din date. Toate regulile și funcțiile de apartenență trebuie definite manual, ceea ce face ca reglarea să fie subiectivă și consumatoare de timp.
- Comportamentul static - Odată proiectat, un sistem fuzzy clasic este fix. Nu se poate adapta la medii noi sau schimbări decât dacă este reconfigurat manual.

- Fără optimizare - Parametrii nu pot fi optimizați intern. Acest lucru duce adesea la performanțe suboptimale, cu excepția cazului în care sunt asistați de metode externe.
- Probleme de scalabilitate - Pentru probleme complexe, de dimensiuni mari, sistemele fuzzy clasice se luptă să se scaleze eficient din cauza bazei de reguli inflexibile și a lipsei de adaptabilitate structurală.
- Dependența de interpretare - Calitatea sistemului depinde în mare măsură de cunoștințele experților, care pot fi inconsistente sau indisponibile, ducând la seturi de reguli părtinitoare sau incomplete.

În concluzie, în timp ce logica fuzzy oferă o modalitate puternică de a gestiona imprecizia, sistemele fuzzy clasice - fără nicio integrare a tehnicilor de învățare sau optimizare - sunt inerent limitate în ceea ce privește scalabilitatea, adaptabilitatea și performanța.

Rețelele neuronale artificiale sunt modele computaționale inspirate de creierul uman, compuse din unități de procesare interconectate numite neuroni. Rețelele neuronale artificiale învață modele și relații în date prin antrenament, de obicei folosind învățare supravegheată sau nesupravegheată. Sunt utilizate pe scară largă pentru clasificare, regresie, prognoză și aproximare a funcțiilor complexe. Atunci când se utilizează RNA singure (fără hibridizare cu alte metode de inteligență artificială, cum ar fi logica fuzzy, optimizarea sau sistemele bazate pe reguli), apar câteva dezavantaje cheie:

- Natura de cutie neagră - RNA nu sunt transparente. Procesul decizional este dificil de interpretat sau explicat, ceea ce este esențial în aplicațiile care necesită responsabilitate sau înțelegere a domeniului.
- Cerințe mari de date - RNA necesită de obicei seturi de date mari, bine etichetate, pentru a se antrena eficient. Cu date limitate sau zgomotoase, performanța lor se poate degrada semnificativ.
- Supra-adaptare - Fără o proiectare și o regularizare atentă, RNA tind să memoreze datele de antrenament, ceea ce duce la o generalizare slabă a intrărilor nevăzute.
- Gestionare deficitară a incertitudinii - RNA clasice nu gestionează în mod inerent imprecizia sau ambiguitatea bine. Acestea presupun intrări și ieșiri precise, ceea ce le face mai puțin potrivite în domenii cu informații vagi sau lingvistice.

- Structură fixă - Odată ce arhitectura este aleasă, aceasta rămâne fixă în timpul antrenamentului. Găsirea structurii potrivite necesită metode de încercare și eroare sau de reglare externă.
- Complexitate computațională - Antrenamentul poate fi costisitor din punct de vedere computațional și consumator de timp, în special pentru rețelele profunde sau recurente, fără o convergență garantată la un minim global.
- Fără cunoștințe de specialitate încorporate - Spre deosebire de sistemele bazate pe reguli, RNA nu încorporează direct cunoștințe de specialitate. Totul trebuie învățat din date, care pot fi ineficiente sau pot omite nuanțe importante specifice domeniului.

În concluzie, deși RNA sunt modele puternice bazate pe date, utilizarea exclusivă a RNA limitează explicabilitatea, interpretabilitatea și robustețea - în special în aplicațiile care implică incertitudine, date limitate sau nevoia de perspectivă umană.

Algoritmii genetici sunt tehnici de căutare și optimizare inspirate de principiile selecției naturale și geneticii. Aceștia dezvoltă o populație de soluții candidate de-a lungul generațiilor folosind operații precum selecția, încrucișarea și mutația. AG sunt utilizați pe scară largă pentru rezolvarea problemelor complexe de optimizare în care soluțiile analitice sunt dificil sau imposibil de obținut. Atunci când se utilizează doar AG, fără a le combina cu alte metode precum rețelele neuronale, sistemele fuzzy sau tehnicile de căutare locală, devin evidente câteva limitări:

- Convergența lentă - AG poate dura multe generații pentru a ajunge la o soluție bună. Fără hibridizare sau îndrumare, acestea converg adesea lent - în special în spații de căutare de dimensiuni mari sau accidentate.
- Convergența prematură - AG clasice sunt predispuse la convergență prematură, unde populația își pierde diversitatea și rămâne prinsă în optime locale, nereușind să exploreze regiuni mai bune ale spațiului de căutare.
- Fără garanție a optimalității - AG sunt euristice și stocastice prin natura lor. Nu garantează găsirea optimului global, ci doar o soluție aproximativă sau „suficient de bună”.
- Sensibilitatea parametrilor - Performanța depinde în mare măsură de parametri precum dimensiunea populației, rata de mutație și rata de

încrucișare. Găsirea combinației potrivite necesită de obicei o ajustare empirică.

- Lipsa cunoștințelor specifice problemei - Analiza genetică este independentă de domeniu. Nu exploatează structura sau cunoștințele specifice problemei decât dacă sunt codificate explicit, ceea ce le poate face ineficiente pentru anumite aplicații.
- Costul computațional - Evaluarea mai multor indivizi pe parcursul mai multor generații, în special dacă fiecare evaluare este costisitoare, poate fi costisitoare din punct de vedere computațional.
- Lipsa memoriei sau a învățării - AG clasice nu rețin informațiile de la rulările anterioare și nu învață din experiențele trecute. Fiecare rulare începe de la zero, ceea ce le face ineficiente în rezolvarea problemelor dinamice sau similare.

În concluzie, deși AG sunt metode puternice de căutare globală, utilizarea AG singulare duce la ineficiențe, sensibilitate la setări și lipsă de adaptabilitate - în special în spații de probleme mari, dinamice sau bogate în cunoștințe.

Se poate concluziona din paragrafele anterioare că utilizarea tehnicilor de IA în abordarea individuală (doar o singură tehnică o dată) este însoțită de multe dezavantaje. Pentru a elimina aceste puncte slabe ale sistemelor clasice (bazate doar pe câte o tehnică de IA), se folosesc sisteme hibride de IA, care sunt rezultatul combinării a două sau mai multor tehnici de IA. Aceste sisteme hibride sunt prezentate în subcapitolele următoare.

### 14.2. Logica fuzzy și rețelele neuronale artificiale

Combinarea logicii fuzzy și a rețelelor neuronale artificiale are ca rezultat sistemele neuro-fuzzy. Aceste tehnici de IA sunt modele hibride puternice care integrează raționamentul uman al sistemelor fuzzy cu capacitățile de învățare și adaptare ale rețelelor neuronale. În continuare se face o descriere succintă a acestei combinații, acoperind motivațiile, tipurile, arhitecturile, metodele de antrenament, avantajele, dezavantajele, implementare și aplicațiile.

Luând în considerare punctele forte ale LF și RNA, combinația celor două duce la sisteme care: învață din date și oferă interpretabilitate parțială precum RNA, dar și gestionează incertitudinea și generează/ajustează automat reguli fuzzy și funcții de apartenență precum LF.

Sistemele neuro-fuzzy se împart în mai multe tipuri, dar cel mai cunoscut și utilizat pe scară largă este sistemul de inferență neuron-fuzzy adaptiv, ANFIS.

***ANFIS***

Sistemul ANFIS a fost introdus pentru prima dată J.S.R. Jang în 1993. Acest sistem neuro-fuzzy are la bază un sistem de inferență fuzzy de tip Sugeno, dar utilizează arhitectura unei rețele neuronale feedforward cu cinci straturi. Caracteristic acestui sistem este faptul că învață reguli fuzzy și parametri ai funcției de apartenență din date folosind un algoritm de propagare înapoi a erorii și estimare prin cele mai mici pătrate.

Arhitectura tipică a unui sistem ANFIS cu două intrări și o ieșire este descrisă în fig. 14.1.:

- Strat 1 – Fuzzificarea intrărilor: Fiecare nod reprezintă o funcție de apartenență (MF) (de exemplu, gaussiană, triunghiulară) pentru o intrare.
- Strat 2 – Puterea de declanșare a regulii: Fiecare nod calculează puterea de declanșare a unei reguli.
- Strat 3 – Normalizare: Fiecare putere de declanșare este normalizată.
- Strat 4 – Consecința regulii: Fiecare nod generează funcția consecventă a unei reguli.
- Strat 5 – Calculul ieșirii: Calculează ieșirea finală ca sumă ponderată a tuturor ieșirilor regulii.

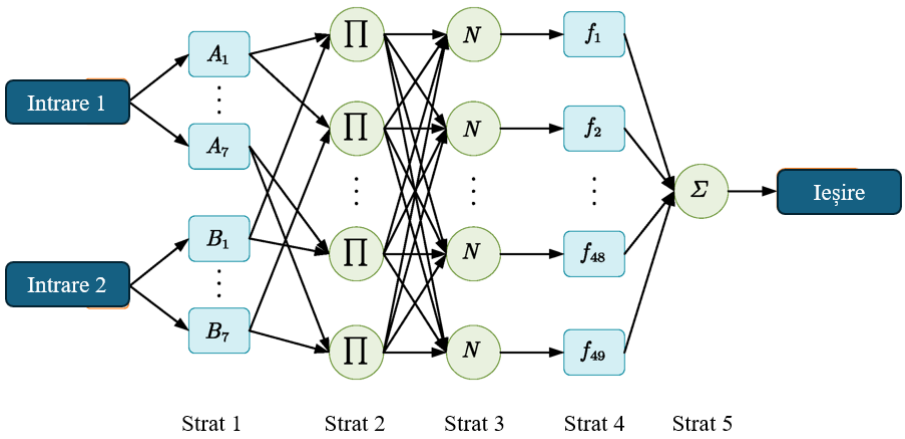



**Fig. 14.1.** Arhitectura de bază a unui sistem ANFIS

Aceste rețele fuzzy au algoritmi speciali de antrenament, precum învățarea hibridă: pase înainte / înapoi (forward pass sau backward pass, ), scăderea gradientului (toți parametrii sunt actualizați folosind propagarea inversă; metoda este mai simplă, dar mai lentă și mai puțin precisă), sau abordări evolutive (utilizate pentru a inițializa sau optimiza parametrii mai global).

Balanța dintre punctele forte și slabe ale acestei tehnici unt ilustrate în figura 14.2.

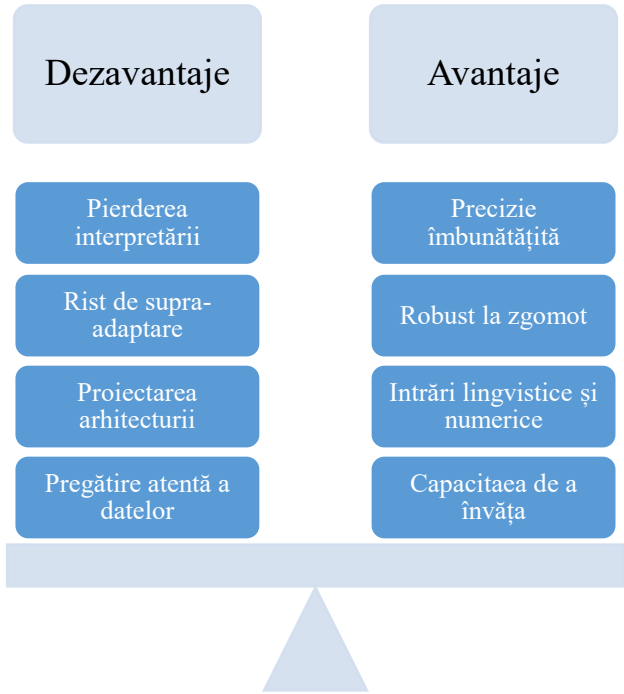



**Fig. 14.2.** Punctele forte și slabe ale ANFIS

Aplicațiile ANFIS în managementul energiei sunt enumerate în tabelul 14.1.

**Tabelul 14.1.** Aplicațiile ANFIS

| **Aplicație** | **Observații** |
|---|---|
| Prognoza sarcinii | Prezicerea cererii de energie electrică pe termen scurt, mediu sau lung.<br>Avantajul neuro-fuzzy: Se ocupă de date de intrare incerte sau imprecise (de exemplu, temperatură, tipul zilei). Învață relații neliniare complexe din datele istorice. Modele utilizate: ANFIS, PANFIS, DENFIS.<br>Impact: Îmbunătățește stabilitatea rețelei, reduce rezerva de rotație, permite o planificare mai bună. |
| Prognoza Energiei Regenerabile | Prognoza Energiei Solare. Intrări: iradiere solară, temperatură, umiditate, ora din zi. Gestionează variabilitatea și intermitența în generarea solară.<br>Prognoza Energiei Eoliene. Intrări: viteza vântului, direcția, presiunea aerului. Modelele neuro-fuzzy |

| | |
|---|---|
| | captează modelele de turbulență neliniară mai bine decât modelele convenționale. |
| Optimizarea consumului de energie în clădirile inteligente | Controlul eficient al sistemelor HVAC, iluminatului și electrocasnicelor.<br>Învață modelele de comportament ale utilizatorilor (prin intermediul RNA). Ia decizii flexibile în condiții de incertitudine (prin reguli fuzzy). Rezultat: Confort sporit, economii de energie și control adaptiv. |
| Managementul Cererii (DSM – Demand Side Management) | Optimizează schimbarea sarcinii, reducerea vârfurilor și ajustările în funcție de timpul de utilizare.<br>Învățați obiceiurile utilizatorilor și utilizarea dispozitivelor. Luați decizii cu logică fuzzy, mai degrabă decât cu praguri fixe. Ajută la aplatizarea curbelor de sarcină și la reducerea facturilor la electricitate. |
| Managementul Energiei Industriale | Monitorizare și optimizare în timp real a consumului de energie în instalațiile de producție și procesare. Exemple de aplicații: Controlul cazanelor, Optimizarea cuptorului, Mentenanță predictivă, sistemele NF se adaptează la condițiile de producție în schimbare. |
| Stabilitatea și controlul sistemului energetic | Controlul puterii reactive, reglarea frecvenței, reglarea AVR. Sisteme neuro-fuzzy: Oferă reglare adaptivă a regulatoarelor. Gestionează perturbațiile sau variațiile de sarcină cu eleganță. Utilizate în FACTS, UPFC și regulatoarele sistemului de excitație. |
| Managementul rețelei inteligente | Echilibrarea sarcinii în timp real, detectarea defecțiunilor și controlul resurselor energetice distribuite. Sistemele neuro-fuzzy integrează: Intrări multiple incerte (vreme, sarcină, semnale de piață). Cunoștințe învățate din comportamentul rețelei. Rezultat: Reziliență, eficiență și fiabilitate mai bune. |
| Managementul sistemulelor de stocare a energiei | Programarea încărcării/descărcării bateriei, Estimarea stării de încărcare (SOC), Gestionarea prin logică neuro-fuzzy a: Incertitudinea în comportamentul bateriei, Programarea adaptivă bazată pe prognoze și tarife. |
| Managementul Energiei în Microrețea | Include generarea locală, stocarea și sarcinile. Sistemele NF sunt utilizate pentru: Prognoza sarcinii, Programarea generării, Dispecerizarea economică în cadrul unei microrețele. Beneficiu cheie: Gestionează |

| | |
|---|---|
| | interacțiuni complexe și incertitudini în modurile insulare sau conectate la rețea. |
| Stabilirea prețurilor la energie și proiectarea tarifelor | Prezice fluctuațiile prețurilor la energie. Gestionează modele dinamice de prețuri bazate pe răspunsul și consumul utilizatorilor. Modelele neuro-fuzzy sunt utilizate pentru a înțelege dinamica neliniară a pieței. |
| Detectarea defecțiunilor și monitorizarea stării | Utilizate în transformatoare, întrerupătoare de circuit și echipamente de rețea. Sistemele neuro-fuzzy învață comportamentul normal și detectează anomalii în condiții de incertitudine. Cheie pentru mentenanța preventivă și fiabilitatea rețelei. |

Celelalte tipuri de sisteme neuron-fuzzy sunt: NEFCON (Neuro-Fuzzy Control – control neuron-fuzzy), FALCON (Fuzzy Adaptive Learning Control Network, rețea de control cu învățare adaptivă), GARIC (Generalized Approximate Reasoning-based Intelligent Control, Control inteligent bazat pe raționament aproximativ generalizat), FINEST (Fuzzy Inference and Neural Network in Estimation, Inferență fuzzy și rețea neuronală în estimare), SONFIN (Self-Constructing Neural Fuzzy Inference Network, Rețea de inferență neuronală fuzzy autoconstructivă), FUN (Fuzzy–Uncertainty Neural Network, Rețea neuronală Fuzzy-Incertitudine), PANFIS (Parsimonious Adaptive Neuro-Fuzzy Inference System, Sistem de inferență neuro-fuzzy parsimonios adaptiv), DENFIS (Dynamic Evolving Neuro-Fuzzy Inference System, Sistem dinamic de inferență neuro-fuzzy în evoluție) și GENEFIS (Generic Evolving Neuro-Fuzzy Inference System, Sistem generic de inferență neuro-fuzzy în evoluție).

Implementarea acestor sisteme se poate face sub formă de software și hardware, așa cum s-a descris în capitolele precedente.

### 14.3. Logica fuzzy și algoritmii genetici

Combinarea logicii fuzzy cu algoritmii genetici creează un sistem hibrid puternic care valorifică punctele forte ale ambelor tehnici. Această hibridizare este denumită în mod obișnuit sistem genetic fuzzy (SGF). În continuare este prezentată o scurtă descriere a acestei combinații, incluzând definiții, motivații, tipuri, metodologii, aplicații și avantaje/dezavantaje.

Sistemele hibride SGF au fost dezvoltate datorită capabilităților sistemelor clasice fuzzy, respectiv genetice. Într-adevăr, logica fuzzy gestionează incertitudinea, imprecizia și vaguitatea folosind reguli lingvistice, iar algoritmii genetici caută și optimizează inspirându-se de evoluția naturală. Însă, sistemele

fuzzy necesită adesea reglarea manuală a funcțiilor de apartenență și a bazelor de reguli, iar algoritmii genetici sunt buni la optimizare, dar nu oferă interpretabilitatea sau luarea deciziilor în timp real în mod direct. Dar, prin combinarea lor, AG optimizează parametrii sau structura sistemelor fuzzy, iar LF oferă sistemelor bazate pe AG o modalitate de a modela raționamentul asemănător celui uman.

În literatura de specialitate sunt prezentate patru tipuri de SFG, în funcție de gradul de implicarea a fiecăreia dintre cele două tehnici de IA:

1. Tipul 1: AG ajustează parametrii unui sistem fuzzy de structură fixă. Astfel, AG optimizează caracteristicile funcțiilor de apartenență, precum forma, lățime, nucleul.
2. Tipul 2: AG evoluează baza de reguli, în timp ce funcțiile de apartenență sunt fixe. Deci, selectează regulile dintr-o bază de reguli stufoasă, sau dezvoltă regulile.
3. Tipul 3: algoritmul genetic dezvoltă atât regulile, cât și funcțiile de membru.
4. Tipul 4: AG dezvoltă întregul sistem fuzzy, inclusiv mecanismele de inferență. Sunt situații în care AG dezvoltă toate caracteristicile sistemului fuzzy, precum definirea regulilor și ajustarea funcțiilor de apartenență.

În aceste sisteme hibride, codificarea folosită în cadrul AG implicați poate fi binară (șirurile de biți reprezintă prezența regulilor, tipul funcției de apartenență sau parametrii), cu valori numerice (numerele reale reprezintă direct parametrii funcțiilor de apartenență), bazată pe arbori (pentru structuri de reguli fuzzy sau arbori de decizie lingvistică) sau mixtă (combină diferite tipuri precum, cu valori numerice pentru funcțiile de apartenență și binară pentru activarea regulilor).

Operatorii genetici utilizați în GFS au atribute diferite, deoarece trebuie să se ia în considerare componentele sistemelor fuzzy. Astfel, în cazul operatorul selecție operează pe baza funcției de adaptare care cuantifică performanța sistemului fuzzy (rata de eroare, precizia clasificării etc.). Încrucișarea presupune combinarea părților a două sisteme fuzzy, de exemplu valorile sau regulile funcțiilor de apartenență. Atunci când se folosește operatorul mutație, pentru a obține noi soluții se introduc variații aleatorii în formele sau regulile funcțiilor de apartenență.

Aceste sisteme hibride pot fi implementate prin folosirea abordărilor prezentate în capitolele anterioare, precum MATLAB, Phyton și LabVIEW.

Avantajele și dezavantajele folosirii acestor sisteme hibride pentru rezolvarea problemelor din energetică sunt sintetizate în fig. 14.3.

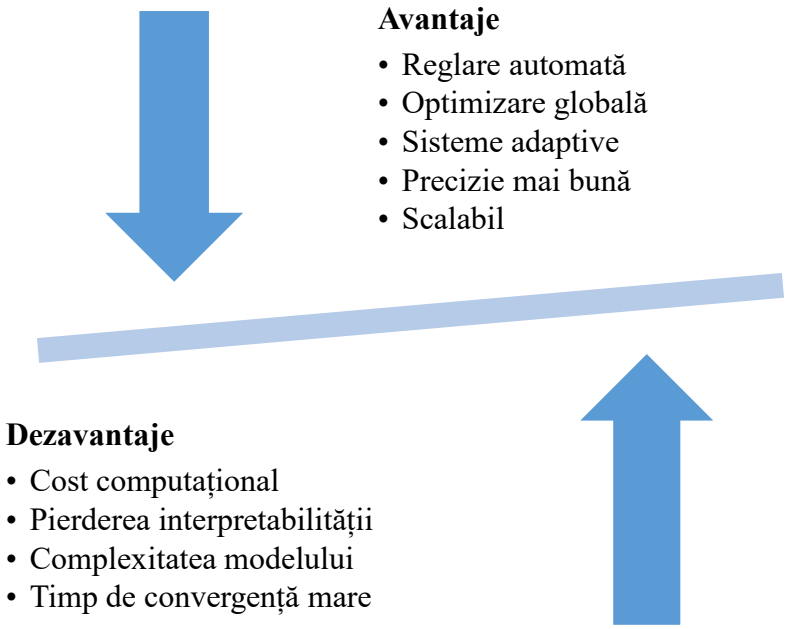



**Fig. 14.3.** Avantajele și dezavantajele SGF

Aceste sisteme hibride sunt din ce în ce mai utilizate în sistemele de management al energiei datorită capacității lor de a gestiona incertitudinea (prin logică fuzzy) și de a optimiza performanța sistemului (prin algoritmi genetici). Mai jos este o listă detaliată a aplicațiilor în managementul energiei, organizate pe categorii, tabelul 14.2.

**Tabelul 14.2.** Aplicațiile SGF

| **Aplicație** | **Observații** |
|---|---|
| Prognoza consumului de energie electrică | Rolul SGF - Sistemele fuzzy modelează relații incerte și neliniare între variabile (temperatură, timp, cerere). AG optimizează regulile fuzzy și funcțiile de apartenență pentru o precizie mai bună. Prognoza consumului pe termen scurt utilizând temperatura și cererea istorică, prognoza conștientă de surse regenerabile, predicția consumului prin managementul cererii. |

| | |
|---|---|
| Programarea și Dispecerizarea Resurselor Energetice | Rolul SGF - Modelele fuzzy surprind imprecizia în comportamentul sarcinii și caracteristicile generatorului. AG optimizează angajamentul unității, dispecerizarea economică și alocarea rezervelor rotative.<br>Dispecerizarea economică a sarcinii luând în considerare constrângerile și pierderile generatorului, problemele de angajament al unității în sistemele de energie termică, dispecerizarea generării de energie regenerabilă (incluzând incertitudinea în producția solară/eoliană). |
| Integrarea Energiei Regenerabile | Rolul SGF - Logica fuzzy ajută la gestionarea incertitudinilor legate de iradierea solară sau viteza vântului. AG ajustează parametrii fuzzy pentru urmărirea punctelor de putere maximă și coordonarea rețelei.<br>MPPT în sistemele fotovoltaice solare folosind reguli fuzzy optimizate prin AG, controlul turbinelor eoliene (controlul unghiului de înclinare, optimizarea puterii reactive), sisteme hibride regenerabile: Managementul bateriei-solar-eolian. |
| Managementul cererii și răspunsul la cerere | Rolul SGF - Logica fuzzy evaluează nivelurile de confort ale consumatorilor și prioritățile aparatelor electrocasnice. AG optimizează programarea aparatelor electrocasnice în funcție de cost și confort.<br>Managementul inteligent al energiei în locuințe cu baze de reguli fuzzy pentru utilizarea aparatelor electrocasnice, optimizarea răspunsului la cerere bazată pe preț (timpul de utilizare, stabilirea prețurilor în timp real), reducerea vârfurilor de consum folosind strategii optimizate de transfer al sarcinii. |
| Sistem de stocare a energiei bateriei | Rolul SGF - Logica fuzzy gestionează incertitudinile legate de starea de încărcare (SoC) și fluctuațiile de sarcină. AG optimizează regulile controlerului fuzzy pentru deciziile de încărcare/descărcare.<br>Programarea stocării energiei în microrețea, gestionarea SoC în vehiculele electrice hibride și bateriile conectate la rețea, gestionarea energiei de rezervă în timpul întreruperilor de alimentare sau a sarcinii mari. |

| | |
|---|---|
| Optimizarea Microrețelelor și a Rețelelor Inteligente | Rolul SGF - Logica fuzzy permite luarea deciziilor multicriteriale (de exemplu, cost, emisii, fiabilitate). GA dezvoltă strategii de control sau optimizează parametrii sistemului în condiții de incertitudine.<br>Controlul micro-rețelelor : Partajarea sarcinii, controlul fluxului de energie, decizii de izolare, coordonarea resurselor energetice distribuite, detectarea defecțiunilor rețelei inteligente și strategii de auto-reparare. |
| Monitorizarea Calității Energiei Electrice și Diagnosticarea Defecțiunilor | Rolul SGF - Logica fuzzy detectează armonicele, căderile/creșterile de tensiune cu praguri imprecise. AG optimizează sensibilitatea și selectivitatea bazei de reguli fuzzy.<br>Clasificarea distorsiunilor armonice, evaluarea stabilității tensiunii, detectarea defecțiunilor transformatoarelor |
| Optimizare multi-obiectiv în sistemele de management energetic | Rolul SGF - logica fuzzy modelează obiective conflictuale (de exemplu, cost vs. fiabilitate vs. emisii). AG efectuează optimizare bazată pe Pareto.<br>Compromisuri cost-emisii în dispecerizarea termică, eficiență energetică vs. confort în sistemele HVAC și funcționare sustenabilă a microrețelelor cu reducerea emisiilor de $CO_2$ |
| Managementul încărcării vehiculelor electrice, VE | Rolul SGF - Logica fuzzy gestionează incertitudinile legate de timpii de sosire, starea bateriei, preferințele utilizatorilor. AG optimizează programele de încărcare și echilibrarea sarcinii.<br>Stații inteligente de încărcare VE cu reguli fuzzy pentru decizii de vârf/în afara orelor de vârf, optimizarea conectării vehiculului la rețea și programarea energiei flotei |
| Sisteme de Management al Energiei Clădirilor | Rolul SGF - Logica fuzzy modelează confortul interior, gradul de ocupare și consumul dinamic de energie. LOGICA optimizează funcționarea HVAC, programele de iluminat și controlul aparatelor.<br>Controlere inteligente HVAC fuzzy, controlul iluminatului cu detectarea prezenței fuzzy și minimizarea consumului de energie în condiții de constrângeri de confort termic. |

### 14.4. Rețele neuronale artificiale și algoritmi genetici

Combinația dintre rețelele neuronale artificiale și algoritmii genetici este o abordare hibridă utilizată pentru a îmbunătăți performanța rețelelor neuronale artificiale prin utilizarea puterii de optimizare a AG. Aceasta este adesea denumită sistem neurogenetic. În următoarele paragrafe se face o descriere succintă a sistemelor neurongenetice, precum și motivația combinării celor două tehnici de IA.

Motivația utilizării RNA împreună cu AG se poate explica prin faptul că:

- RNA sunt modele puternice de învățare, dar adesea suferă de blocarea în minimele locale atunci când se utilizează metode tradiționale de antrenament, cum ar fi coborârea în gradient. Plus, acestea au sensibilitate la valorile inițiale ale ponderilor și dificultăți cu convergența lentă în unele cazuri.
- Algoritmii genetici pot ajuta la depășirea acestor probleme prin furnizarea unei metode de căutare globală pentru găsirea unor ponderi inițiale sau structuri de rețea mai bune. Mai mult, se poate evita blocarea în minime locale prin diversitate evolutivă, și gestionarea eficientă a suprafețelor de eroare complexe sau discontinue.

Rețelele neuronale și algoritmii genetici se poat combina în trei metode, ducând la dezvoltarea de teri tipuri de sisteme neurogenetice.

Primul tip de sisteme neurogenetice se obține prin implicarea AG ân procesul de ajustare a ponderilor RNA. Astfel, AG este utilizat pentru a evolua ponderile conexiunilor rețelei neuronale în loc să utilizeze retropropagarea. În cadrul algoritmului, fiecare cromozom reprezintă un set complet de ponderi ale RNA. Funcția de fitness definită pentru AG se bazează pe eroarea RNA în procesul de testare / validare (de exemplu, eroarea medie pătratică pe un set de validare).

Al doilea tip de sisteme neurogenetice presupune utilizarea AG pentru optimizarea structurii rețelei neuronale. Caracteristic acestui tip de sisteme hibride sunt următoarele: (1) AG optimizează topologia rețelei: numărul de straturi, neuroni, conexiuni, (2) cromozomii codifică structura rețelei (de exemplu, șiruri binare unde 1 = conexiune activă, 0 = inactiv). Ultima abordare poartă denumirea de neuroevoluție.

Al treilea tip de sisteme neurogenetice se obțin prin utilizarea AG pentru selecția caracteristicilor / optimizarea intrării. Întrucât AG selectează cel mai bun subset de caracteristici de intrare pentru a fi utilizat în antrenarea RNA, acest fapt

poate îmbunătăți performanța și reduce supraadaptarea, în special cu seturi de date mari sau zgomotoase.

În sistemele neurogenetice, reprezentarea soluțiilor se face prin codificarea binară, cu valori numerice sau hibridă. Codificarea binară se folosește pentru a reprezenta structura rețelei sau valorile discrete cu care lucrează aceasta. Codificarea cu valori numerice este utilizată pentru a reprezenta valorile ponderilor care sunt alelele cromozomilor. În sfârșit, codificarea hibridă presupune implicarea ambelor tipuri de codificare, binară și valori numerice.

Avantajele utilizării sistemelor neurogenetice sunt următoarele:

1. Posibilitatea găsiri de soluții globale mai bune decât metodele bazate pe gradient.
2. Flexibilitate și adaptabilitate la funcțiile de fitness personalizate.
3. Utilitate în problemele care prezintă multe zgomote sau elemente neliniare (de exemplu, sisteme de alimentare, procesare a semnalelor, recunoaștere a modelelor).

Contrar, provocăriile utilizării acestor sisteme se referă la următoarele aspecte:

- Aceste sisteme hibride sunt mai scumpe din punct de vedere computațional din cauza numeroaselor evaluări.
- Sistemele neurogenetice sunt mai lente decât antrenamentul tradițional pentru probleme mici sau simple.
- Dezvoltarea unui sistem neurogenetic necesită o proiectare atentă a cromozomilor și a funcției de adaptare.

Combinația RNA cu AG are mai multe aplicații valoroase în sistemele de management al energiei. Aceste modele hibride sunt deosebit de utile în optimizarea, prognozarea și controlul resurselor energetice atât în sistemele convenționale, cât și în cele de rețele inteligente. Mai jos sunt prezentate aplicațiile cheie în managementul energiei în care sistemele neuogenetici joacă un rol vital, tabelul 14.3.

**Tabelul 14.3.** Aplicațiile sistemelor neurogenetice

| **Aplicație** | **Observații** |
|---|---|
| Prognoza sarcinii | Scop - Prezicerea cererii viitoare de energie pe termen scurt, mediu sau lung. RNA învață modelele de cerere |

| | |
|---|---|
| | din datele istorice. AG optimizează ponderile RNA sau selectează cele mai bune caracteristici de intrare.<br>Predicția zilnică sau orară a sarcinii în rețele inteligente, microrețele și utilități. |
| Prognoza Energiei Regenerabile | Scop - Prezicerea generării de energie din surse regenerabile, cum ar fi solara și eoliana. RNA modelează relația neliniară dintre variabilele de mediu (iradiere, temperatură, viteza vântului) și producție. AG îmbunătățește precizia predicției prin reglarea parametrilor RNA sau selectarea celor mai bune variabile de intrare.<br>Prognoza producției fotovoltaice în parcurile solare sau generarea de energie eoliană pentru a gestiona stabilitatea rețelei. |
| Optimizarea consumului de energie | Scop: Reducerea costurilor cu energia sau a cererii maxime în configurațiile rezidențiale, comerciale sau industriale. RNA prezice comportamentul de utilizare a energiei. AG optimizează programele aparatelor sau strategiile de control.<br>Case sau clădiri inteligente care gestionează utilizarea HVAC, iluminatului și echipamentelor pentru a minimiza facturile la electricitate. |
| Managementul cererii (demand side management) | Scop: Încurajarea utilizatorilor să modifice consumul în funcție de preț sau de condițiile de sarcină. RNA prezice răspunsul utilizatorului și impactul asupra sarcinii. AG găsește cele mai bune stimulente sau strategii de control.<br>Strategii de stabilire a prețurilor în funcție de timpul de utilizare, schimbarea sarcinii sau reducerea vârfurilor de putere în rețelele inteligente. |
| Flux Optim de Putere (OPF) | Scop: Minimizarea costurilor, pierderilor sau emisiilor, respectând în același timp constrângerile sistemului energetic. RNA pot fi antrenate să estimeze parametrii optimi. AG este utilizat pentru a optimiza ieșirile generatoarelor, tensiunile și fluxurile de putere.<br>Control în timp real în centrele de dispecerizare a energiei pentru a reduce costurile de operare și a asigura fiabilitatea rețelei. |

| | |
|---|---|
| Managementul Sistemului de Stocare a Energiei în Baterii | Scop: Încărcarea/descărcarea eficientă a bateriilor în microrețele sau în configurații regenerabile. RNA prognozează sarcina și generarea. AG determină programe optime de încărcare sau strategii SoC (Stare of Charge - Stare de Încărcare).<br>Sisteme solare + baterii pentru stocarea energiei la domiciliu sau în comunitate. |
| Prețurile Energiei și Licitarea pe Piață | Scop: Prezicerea prețurilor pieței energiei electrice și optimizarea strategiilor de licitare. RNA învață tendințele prețurilor. AG găsește strategia optimă de licitare sau programul de angajament.<br>Furnizori de energie și prosumatori pe piețele dereglementate de energie electrică. |
| Optimizarea Eficienței Energetice Industriale | Scop: Reducerea consumului de energie în sistemele industriale mari.<br>RNA modelează fluxuri complexe de energie. AG identifică setări optime sau parametri de proces.<br>Optimizarea sarcinilor motoarelor, sistemelor de aer comprimat, sistemelor de încălzire și răcire. |

În implementarea acestor sisteme neurogenetice se pot folosi aceleași abordări ca în cazul sistemelor singulare, adică implicarea limbajelor și mediilor de programare dedicate și a bibliotecilor predefinite: MATLAB, Phyton, LabVIEW, C/C++ etc.; precum utilizarea dispozitivelor fizice dedicate: FPGA, microcontrolere etc.

### 14.5. Concluzii

Acest suport de curs s-a focalizat pe cele trei tehnici de IA ”seniore”, și anume logica fuzzy, rețelele neuronale artificiale și algoritmii genetici. Cele trei tehnici se caracterizează prin următoarele:

- Logică fuzzy gestionează incertitudinii și folosește variabile lingvistice caracteristice sistemelor energetice.
- Rețele neuronale artificiale învață din date pentru predicție, clasificare și optimizare.
- Algoritmi genetici realizează o căutare euristică a soluțiilor optime în probleme complexe, neliniare.
- Obiectivul comun a acestor tehnici este îmbunătățirea luării deciziilor, a controlului și a optimizării în sistemele energetice.

O analiză a principalelor aplicații ale celor trei tehnici de IA arată că LF este folosită cu succes în control și prognoză, RNA sunt implicate în special în probleme de predicție și clasificare, iar AG dau rezultate foarte bune în programarea consumului și alte probleme de optimizare.

Aceste tehnici de IA se pot folosi cu succes dacă se îndeplinesc câteva condiții esențiale. Astfel, sistemele fuzzy pot să fie dezvoltate cu succes, atunci când sunt disponibile cunoștințe de specialitate, dar datele sunt limitate. În cazul rețelelor neuronale artificiale, principala condiție este disponibilitateaa unor bogate baze de date istorice, care vor sta la baza modelelor învățate. Algoritmii genetici se utilizează pentru probleme de optimizare cu spații de căutare mari și complexe. Combinarea acestor tehnici conduce la sisteme hibride a căror rezultate sunt net superioare sistemelor clasice.

La final se pot trage următoarele concluzii cu privire la utilizarea tehnicilor de IA în managementul energiei electrice:

- Adaptabilitatea - Aceste tehnici se adaptează la problemele de management al energiei din sistemele electroenergetice, de exemplu la schimbarea cererii de energie și la dinamica pieței.
- Eficiența - Metodele care implică utilizarea acestor tehnici de IA îmbunătățesc eficiența în generarea, transport, distribuția și consumul de energie.
- Scalabilitatea – Aplicațiile dezvoltate pe baza tehnicilor de IA pot fi aplicate la diferite niveluri de mărime a sistemelor energetice, de la microrețea la nivelul rețelei naționale.
- Interdisciplinaritatea – Aplicațiile care au la bază IA presupun combinarea cunoștințelor despre sistemul energetic, și cunoștințe despre tehnicile de IA implicate.

Tendințele viitoare în utilizarea tehnicilor de IA în managementul energiei sunt enumerate în continuare:

1. Integrare cu IoT (Internet-of-Things) și rețele inteligente.
2. Gestionarea energiei în timp real cu inteligență artificială încorporată.
3. Controlul stocării energiei folosind LF, RNA, sau AG.
4. IA explicabilă în sistemele energetice.

Pe măsură ce sistemele energetice devin mai inteligente, rolul algoritmilor inteligenți sau care au la bază tehnici de IA devine mai important. Logica fuzzy,

rețelele neuronale și algoritmii genetici nu sunt doar instrumente, cu sunt căi către sisteme energetice sustenabile, eficiente și autonome.

În final câteva aplicații practice ale LF, RNA și AG care sunt folosite în mod activ în producție (lumea reală), ci doar ân cercetare, în diferite țări în sistemele lor electroenergetice.

***Regatul Unit***

ScottishPower – Sisteme de detectare și diagnosticare a defecțiunilor, folosind LF (link: https://www.spenergynetworks.co.uk/news/pages/demonstrating_worlds_first_real_time_fault_level_measurement_system.aspx).

National Grid UK, Scottish and Southern Electricity Networks. Aplicații: Prognoza sarcinii pe termen scurt folosind modele RNA antrenate pe baza datelor istorice de sarcină, vreme și evenimente și Detectarea defecțiunilor la cablurile subterane și transformatoare folosind clasificatori RNA antrenați), (link: https://ssen-innovation.co.uk/nia-projects/nia-ssen-0051-synaps-2-fault-detection-classification-location-solution/ ).

***SUA***

Pacific Gas & Electric (PG&E), EPRI și utilități prin programe finanțate de DOE. Aplicații: RNA utilizat în mentenanța predictivă pentru transformatoare și întrerupătoare de circuit și Clasificarea calității energiei electrice folosind modele wavelet + RNA pentru a detecta căderi de tensiune, umflături, tranzitorii.

ISO New England, PJM (prin furnizori precum GE și ABB). Aplicație: AG utilizat pentru angajamentul unitar și dispecerizarea economică în condiții de constrângeri neconvexe (costuri de pornire/oprire, marje de rezervă). (link: https://www.iso-ne.com/committees/key-projects/implemented/new-englands-future-grid-initiative-key-project ).

Southern California Edison, Sandia National Labs. Descriere: Logica fuzzy a fost integrată în releele substațiilor pentru protecție adaptivă (proiecte Sandia, EPRI). În clădiri: Controlul fuzzy este utilizat pentru răspunsul la cerere HVAC (proiecte California DR). (link: https://transmission.epri.com/p37_substations/ ).

***Germania***

Siemens, RWTH Aachen. Aplicații: AG utilizat pentru optimizarea programelor unităților de cogenerare (CHP). De asemenea, utilizat în studiile de planificare a transportului pentru foaia de parcurs Energiewende a Germaniei.

50Hertz, E.ON. Aplicație: Modele de prognoză bazate pe RNA pentru generarea de energie eoliană la nivel de rețea. (link: https://publica-

rest.fraunhofer.de/server/api/core/bitstreams/1cc7d355-27ad-4758-80f5-4fb0aa7cb03d/content ).

Siemens R&D și RWTH Aachen. Aplicație: Analiza genetică optimizează fluxurile de energie în microrețelele industriale, luând în considerare semnalele de preț, emisiile de carbon și generarea locală. Implementat în demonstrații de microrețele în campusuri și în industrie.

Siemens și Institutul Fraunhofer. Descriere: Controlerele fuzzy logic au fost implementate pentru controlul pasului turbinelor eoliene și sistemele de asistență a rețelei pentru a îmbunătăți calitatea energiei și răspunsul în frecvență în condiții de penetrare ridicată a energiei regenerabile.

***Japonia***

TEPCO, Hitachi. Aplicație: RNA utilizată pentru a analiza datele contoarelor inteligente pentru detectarea anomaliilor, recunoașterea modelelor de sarcină și predicția cererii. Implementat în programele de rețele inteligente din zona metropolitană Tokyo.

TEPCO și în proiectul microrețea Sendai. Descriere: Controlerele fuzzy sunt utilizate pentru a gestiona stocarea bateriilor, a prioritiza deconectarea de sarcină și a echilibra între sursele fotovoltaice, de rețea și de rezervă diesel. Rezultatul acestui sistem - Microrețeaua a menținut funcționarea insulară după cutremurul (2011), în parte datorită logicii sale decizionale inteligente bazate pe LF.

***India***

Power Grid Corporation of India (PGCIL) în platforme de testare regionale. Descriere: Sisteme de control adaptiv fuzzy-PI și fuzzy pentru controlul fluxului de putere în frecvență și linie de legătură în rețelele electrice din nord și vest.

NTPC, Corporația de Energie Solară din India (SECI). Aplicație: RNA utilizată pentru predicția radiației solare și a energiei fotovoltaice în ferme solare mari.

Centre regionale de dispecerizare și proiecte universitate-industrie. Aplicație: Optimizarea ELD în timp real folosind AG, luând în considerare efectele punctului de supapă și constrângerile generatorului. Utilizat în proiecte pilot la centrale termice în rețelele NTPC și de stat.

***Brazilia***

CPFL Energia și universități. Aplicație: Clasificatori bazați pe RNA utilizați în sistemele de generare distribuită pentru a detecta evenimente de insulă neintenționate (critice pentru siguranță). Testat pe teren în programe pilot solare și eoliene.

Elektro, Eletrobras. Aplicație: Analiza genetică este aplicată la reconfigurarea optimă a rețelelor de distribuție pentru a minimiza pierderile tehnice. Testat pe teren cu date SCADA și intrări de sarcină în timp real.

Eletrobras și consorții universitate-industrie. Descriere: FL utilizat pentru estimarea pierderilor, controlul bateriilor de condensatoare și reglarea tensiunii în rețelele rurale. Testat pe teren în zone cu provocări ridicate legate de furturi și căderi de tensiune.

***China***

State Grid China. Aplicație: RNA utilizat pentru profilarea sarcinii clienților și clasificarea stării transformatoarelor folosind date de senzori online.

CEPRI și agențiile majore de planificare a rețelelor. Aplicație: Analiza genetică este utilizată pentru planificarea extinderii rețelei de transport și amplasarea dispozitivelor FACTS.

State Grid Corporation of China (SGCC). Descriere: Logica fuzzy este utilizată în controlerele stațiilor de conversie pentru a îmbunătăți robustețea liniilor UHVDC. De asemenea, utilizată în parcurile eoliene din Mongolia Interioară pentru a îmbunătăți controlul ratei de creștere și conformitatea cu rețeaua.

***Coreea de Sud***

KEPCO și Jeju Island Smart Grid Pilot. Descriere: Logica fuzzy este încorporată în sistemele de management al energiei casnice (HEMS) și în controlerele de energie ale clădirilor pentru a lua decizii în timp real în condiții de incertitudine privind prețul și vremea. (link: https://home.kepco.co.kr/kepco/ indi/foreign /en/html/B/E/ENBEHP002.html).

***Italia***

ENEL (companie de utilități italiană). Descriere: Sisteme cu logică fuzzy aplicate în backend-urile de contorizare inteligentă pentru a gestiona modele de sarcină ambigue pentru clienții de joasă tensiune și pentru a detecta nereguli (cum ar fi pierderile non-tehnice).

ENEL. Aplicații: Prognoza pe termen scurt a sarcinii folosind modele RNA integrate în sistemul de management al energiei ENEL. De asemenea, utilizat în instrumente de monitorizare a calității energiei bazate pe inteligență artificială. (link: https://openinnovability.enel.com/stories/articles/2022/04/mystai-powering-sustainable-ai ).

ENEL Distribuzione. Aplicații: Optimizarea bazată pe analiză genetică a topologiei rețelei de distribuție pentru a reduce pierderile și a îmbunătăți

profilurile de tensiune. Integrare: În platforma lor de sistem avansat de management al distribuției (ADMS).

***Franța***

RTE (Réseau de Transport d'Électricité). Aplicații: Clasificarea modelelor de sarcină și detectarea pierderilor non-tehnice prin RNA. Integrare cu analize backend de contorizare inteligentă (în parteneriat cu EDF).

EDF și RTE. Aplicații: AG utilizat pentru optimizarea proiectării, amplasării și programării întreținerii predictive a substațiilor în condiții de incertitudine.

***Spania***

Iberdrola și Red Eléctrica de España (REE). Aplicații: Rețele neuronale antrenate pentru clasificarea și localizarea defecțiunilor în liniile de distribuție de medie și joasă tensiune. Utilizare reală: Implementat în centre de control regionale ca parte a modernizării rețelei.

***Țările de Jos***

Alliander. Aplicații: Modele bazate pe RNA pentru a prezice congestia rețelei în zonele cu adoptare ridicată a vehiculelor electrice și a energiei solare. (link: https://lfenergy.org/openstef-delivers-operational-forecasts-at-hundreds-of-grid-locations-for-alliander/ ).

***Suedia***

Vattenfall. Aplicații: RNA utilizată pentru predicția energiei și controlul stocării în platforme de testare a microrețelelor la distanță (de exemplu, regiuni arctice). (link: https://group.vattenfall.com/press-and-media/newsroom/2025/microgrid-lets-paradise-island-arholma-cast-off-from-the-main-land).

***Finlanda***

Fortum și VTT (Centrul de Cercetare Tehnică din Finlanda). Aplicații:

Analog genetic utilizat în optimizarea multi-obiectiv a participării flexibile la sarcină pe piețele în timp real. (link: https://www.fortum.com/products-and-services/smart-energy-solutions/fortum-spring ).

# Bibliografie


[1]. Schank R.C., *What is Artificial Inteligence, Anyway ?*, AI Magazine, Vol. 8, No. 4, 1987.

[2]. Mircea Eremia et.al, T*ehnici de Inteligenţă Artificială. Concepte şi aplicaţii în sistemele electroenergetice, Cap. 1*, Ed. AGIR, Bucureşti, 2001.

[3]. Mihai Gavrilaş, *Inteligenţa Artificială şi Aplicaţii în Energetică, Cap. 1*, Vol I, Ed. Gh. Asachi, Iaşi, 2002.

[4]. Şerban Gabriela, Pop H.F., *Tehnici de Inteligenţă Artificială. Abordări bazate pe agenţi inteligenţi, Cap. 1*, Ed. Mediamira, Cluj-Napoca, 2004.

[5]. Chindriş M., Cziker A., *Utilizarea logicii fuzzy în energetică, Cap. 1*, Ed. Casa Cărţii de Ştiinţă, Cluj-Napoca, 2004.

[6]. Takagi, T., & Sugeno, M. *Fuzzy identification of systems and its applications to modeling and control.* IEEE Transactions on Systems, Man, and Cybernetics, 15(1), 116–132, 1985.

[7]. Ross, T. J. *Fuzzy Logic with Engineering Applications (3rd ed.).* Wiley, 2010.

[8]. Jang, J. S. R., Sun, C. T., & Mizutani, E. *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence.* Prentice Hall, 1997.

[9]. Klir, G. J., & Yuan, B. *Fuzzy Sets and Fuzzy Logic: Theory and Applications.* Prentice Hall, 1995.

[10]. Pedrycz, W., & Gomide, F. *Fuzzy Systems Engineering: Toward Human-Centric Computing.* Wiley-IEEE Press, 2007.

[11]. Mendel, J. M. *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions.* Prentice Hall, 2001.

[12]. Mendel, J. M., & John, R. I. *Type-2 fuzzy sets made simple.* IEEE Transactions on Fuzzy Systems, 10(2), 117–127, 2002.

[13]. Hagras, H. *Type-2 FLCs: A new generation of fuzzy controllers.* IEEE Computational Intelligence Magazine, 2(1), 30–43, 2007.

[14]. Wagner, C., & Hagras, H. *Toward general type-2 fuzzy logic systems based on zSlices.* IEEE Transactions on Fuzzy Systems, 18(4), 637–660, 2010.

[15]. Mendel, J. M. *An Introduction to Type-2 Fuzzy Logic Theory and Applications.* Springer, 2017.

[16]. Matsumoto, T. *Fuzzy systems and their applications to control and decision-making problems.* Proceedings of the International Conference on Fuzzy Logic & Applications, 1988.

[17]. Matsumoto, T. *Hierarchical fuzzy control and self-organizing fuzzy systems.* Journal of Advanced Computational Intelligence, 1(2), 65–72, 1993.

[18]. Matsumoto, T. & Sugeno, M. *A study on adaptive fuzzy inference systems.* IEEE International Conference on Fuzzy Systems, 1995.

[19]. Yager, R. R., & Filev, D. P. *Essentials of Fuzzy Modeling and Control.* Wiley, 1994.

[20]. Pedrycz, W. *Fuzzy Systems: Modeling and Control.* Wiley, 2012.

[21]. R. Bellman and L.A. Zadeh, Decision-Making in a Fuzzy Environment, 1970.

[22]. C. Huang, C. Lee, and M. Wang, A Fuzzy Decision Support System for Risk Analysis in E-Commerce Development, 2004.

[23]. M. G. Romero et al., Fuzzy Logic–Based Clinical Decision Support System for the Evaluation of Renal Function in Posttransplant Patients, 2020.

[24]. A.K. Jain, M.N. Murty, and P.J. Flynn, A Survey of Fuzzy Clustering, 1999.

[25]. P. Jajuga, Fuzzy Cluster Analysis from the Viewpoint of Robust Statistics, 2008.

[26]. M. H. Saadat et al., A Fuzzy Clustering Approach to Identify Pedestrians' Traffic Behavior Patterns, 2023.

[27]. Alves de Araujo Junior, C.A.; Mauricio Villanueva, J.M.; Almeida, R.J.S.d.; Azevedo de Medeiros, I.E. Digital Twins of the Water Cooling System in a Power Plant Based on Fuzzy Logic. *Sensors* 2021, *21*, 6737. https://doi.org/10.3390/s21206737.

[28]. Magaji, N.; Mustafa, M.W.B.; Lawan, A.U.; Tukur, A.; Abdullahi, I.; Marwan, M. Application of Type 2 Fuzzy for Maximum Power Point Tracker for Photovoltaic System. *Processes* 2022, *10*, 1530. https://doi.org/ 10.3390/pr10081530.

[29]. Rocha, P.H.A., Coelho, F. Fuzzy control for automatic operation of bank capacitors installed in a substation. *Electr Eng* 104, 4357–4366 (2022). https://doi.org/10.1007/s00202-022-01624-2.

[30]. Khampariya, P.; Panda, S.; Alharbi, H.; Abdelaziz, A.Y.; Ghoneim, S.S.M. Coordinated Design of Type-2 Fuzzy Lead–Lag-Structured SSSCs and PSSs for Power System Stability Improvement. *Sustainability* 2022, *14*, 6656. https://doi.org/10.3390/su14116656.

[31]. Rikos, E.; Caerts, C.; Cabiati, M.; Syed, M.; Burt, G. Adaptive Fuzzy Control for Power-Frequency Characteristic Regulation in High-RES Power Systems. *Energies* 2017, *10*, 982. https://doi.org/10.3390/en10070982.

[32]. Dayana Andrade-Benavides, Diego Vallejo-Huanga, Paulina Morillo, Fuzzy Logic Model for Failure Analysis in Electric Power Distribution Systems, Procedia Computer Science, Volume 204, 2022, Pages 497-504, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2022.08.061.

[33]. Q. -U. Ain, S. Iqbal and H. Mukhtar, Improving Quality of Experience Using Fuzzy Controller for Smart Homes, *IEEE Access*, vol. 10, pp. 11892-11908, 2022, doi: 10.1109/ACCESS.2021.3096208.

[34]. Tavarov, S.S.; Matrenin, P.; Safaraliev, M.; Senyuk, M.; Beryozkina, S.; Zicmane, I. Forecasting of Electricity Consumption by Household Consumers Using Fuzzy Logic Based on the Development Plan of the Power System of the Republic of Tajikistan. *Sustainability* 2023, *15*, 3725. https://doi.org/10.3390/su15043725.

[35]. Lanjing Xu, Zhiwei Wang, Yanfeng Liu, Lin Xing, Energy allocation strategy based on fuzzy control considering optimal decision boundaries of standalone hybrid energy systems, Journal of Cleaner Production, Volume 279, 2021, 123810, ISSN 0959-6526, https://doi.org/10.1016/j.jclepro .2020.123810.

[36]. https://pabloinsente.github.io/the-adaline, ultima accesare martie 2025.

[37]. Paulo Botelho Pires, Inês Veiga Pereira, and José Duarte Santos, Artificial Neural Networks: History and State of the Art, 2023, IGI Global.

[38]. Mohamad H. Hassoun, Fundamentals of Artificial Neural Networks, 1995, Google Books, MIT Press.

[39]. M. M. Hammad, Artificial Neural Network and Deep Learning: Fundamentals and Theory, 2024.

[40]. Michael A. Nielsen, Neural Networks and Deep Learning, 2015, Free computer books.

[41]. https://www.almabetter.com/bytes/articles/convolutional-neural-networks, accesat ultim adată 04.04. 2025.

[42]. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. [Chapter 10]

[43]. Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2), 179–211.

[44]. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.

[45]. Olah, C. (2015). Understanding LSTM Networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/
[46]. Cho, K. et al. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *arXiv:1406.1078*.
[47]. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.
[48]. Kohonen, T. (2001). *Self-Organizing Maps*. Springer Series in Information Sciences.
[49]. Maass, W. (1997). Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9), 1659–1671.
[50]. Alamin, Y.I.; Anaty, M.K.; Álvarez Hervás, J.D.; Bouziane, K.; Pérez García, M.; Yaagoubi, R.; Castilla, M.d.M.; Belkasmi, M.; Aggour, M. Very Short-Term Power Forecasting of High Concentrator Photovoltaic Power Facility by Implementing Artificial Neural Network. Energies 2020, *13*, 3493. https://doi.org/10.3390/en13133493.
[51]. Arferiandi, Y.D.; Caesarendra, W.; Nugraha, H. Heat Rate Prediction of Combined Cycle Power Plant Using an Artificial Neural Network (ANN) Method. Sensors 2021, 21, 1022. https://doi.org/10.3390/s21041022.
[52]. Al-Majidi, S.D.; Kh. AL-Nussairi, M.; Mohammed, A.J.; Dakhil, A.M.; Abbod, M.F.; Al-Raweshidy, H.S. Design of a Load Frequency Controller Based on an Optimal Neural Network. Energies 2022, 15, 6223. https://doi.org/10.3390/en15176223.
[53]. Mahar, H.; Munir, H.M.; Soomro, J.B.; Akhtar, F.; Hussain, R.; Elnaggar, M.F.; Kamel, S.; Guerrero, J.M. Implementation of ANN Controller Based UPQC Integrated with Microgrid. Mathematics 2022, 10, 1989. https://doi.org/10.3390/math10121989.
[54]. Irfan, M.M.; Malaji, S.; Patsa, C.; Rangarajan, S.S.; Hussain, S.M.S. Control of DSTATCOM Using ANN-BP Algorithm for the Grid Connected Wind Energy System. Energies 2022, 15, 6988. https://doi.org/10.3390/en15196988.
[55]. Waseem, M.; Lin, Z.; Yang, L. Data-Driven Load Forecasting of Air Conditioners for Demand Response Using Levenberg–Marquardt Algorithm-Based ANN. Big Data Cogn. Comput. 2019, 3, 36. https://doi.org/10.3390/bdcc3030036.

[56]. Kim, T.-G.; Lee, H.; An, C.-G.; Yi, J.; Won, C.-Y. Hybrid AC/DC Microgrid Energy Management Strategy Based on Two-Step ANN. *Energies* 2023, *16*, 1787. https://doi.org/10.3390/en16041787.

[57]. Kim, S.; Jung, D.-Y. Fault Estimation of Rack-Driving Motor in Electrical Power Steering System Using an Artificial Neural Network Observer. Electronics 2022, 11, 4149. https://doi.org/10.3390/electronics11244149.

[58]. Abrantes, G.C.; Costa, V.S.; Perdigão, M.S.; Cruz, S. Mutual Inductance Estimation Using an ANN for Inductive Power Transfer in EV Charging Applications. Energies 2024, 17, 1615. https://doi.org/10.3390/en17071615.

[59]. John H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, 1992, ISBN-13: 978-026-2581-110, MIT Press.

[60]. David E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, 1989, ISBN-13: 978-020-1157-673, Addison-Wesley.

[61]. Melanie Mitchell, An Introduction to Genetic Algorithms, 1996, ISBN-13: 978-026-2631-853, MIT Press.

[62]. A.E. Eiben & J.E. Smith, Introduction to Evolutionary Computing, 2016 (Softcover reprint of the original 2nd ed. 2015), ISBN-13: 978-3662499856, Springer.

[63]. T. Bäck, D.B. Fogel, & Z. Michalewicz (Editors), Handbook of Evolutionary Computation, 1997, ISBN-13: 978-0750308953, IOP Publishing Ltd.

[64]. M.A. Abido, Optimal Power Flow using Evolutionary Algorithms, 2002, ISSN: 0885-8950, Journal: IEEE Transactions on Power Systems, Vol. 17, No. 2, pp. 519–527.

[65]. L.L. Lai & J.T. Ma, Application of Genetic Algorithms to Reactive Power Planning—Comparison with Nonlinear Programming Approach, 1997, ISSN: 0885-8950, Journal: IEEE Transactions on Power Systems, Vol. 12, No. 1, pp. 198–206.

[66]. K. Nara et al., Practical Distribution System Loss Minimization by Genetic Algorithm, 1996, ISSN: 0885-8977, Journal: IEEE Transactions on Power Delivery, Vol. 11, No. 1, pp. 390–395.

[67]. K.T. Chaturvedi, M. Pandit, & L. Srivastava, Self-Adaptive Differential Evolution for Optimal Power Flow, 2008, ISSN: 0196-8904, Journal: Energy Conversion and Management, Vol. 49, No. 4, pp. 1064–1071.

[68]. Brennenstuhl M, Otto R, Pietruschka D, Schembera B, Eicker U. Optimized Dimensioning and Economic Assessment of Decentralized Hybrid Small Wind and Photovoltaic Power Systems for Residential Buildings. Energies. 2025; 18(7):1811. https://doi.org/10.3390/en18071811.

[69]. Gospodinova E, Nenov D. Forecasting Models and Genetic Algorithms for Researching and Designing Photovoltaic Systems to Deliver Autonomous Power Supply for Residential Consumers. *Applied Sciences*. 2025; 15(9):5033. https://doi.org/10.3390/app15095033.

[70]. Silva CH, Mendes SFS, Garcés Negrete LP, López-Lezama JM, Muñoz-Galeano N. Optimizing Photovoltaic Generation Placement and Sizing Using Evolutionary Strategies Under Spatial Constraints. *Energies*. 2025; 18(8):2091. https://doi.org/10.3390/en18082091.

[71]. Ji C, Nie X, Chen S, Cheng M, Dai Z. Optimization of a Nuclear–CSP Hybrid Energy System Through Multi-Objective Evolutionary Algorithms. Energies. 2025; 18(9):2189. https://doi.org/10.3390/en18092189.

[72]. ] Shi T, Wang C, Chen Z. The Multi-Point Cooperative Control Strategy for Electrode Boilers Supporting Grid Frequency Regulation. Processes. 2025; 13(3):785. https://doi.org/10.3390/pr13030785.

[73]. Urrea-Aguirre C, Saldarriaga-Zuluaga SD, Bustamante-Mesa S, López-Lezama JM, Muñoz-Galeano N. Optimal Placement and Sizing of Modular Series Static Synchronous Compensators (M-SSSCs) for Enhanced Transmission Line Loadability, Loss Reduction, and Stability Improvement. Processes. 2025; 13(1):34. https://doi.org/10.3390/pr13010034.

[74]. Zhan J, Huang M, Sun X, Chen Z, Zhang Z, Li Y, Zhang Y, Ai Q. Coordinated Interaction Strategy of User-Side EV Charging Piles for Distribution Network Power Stability. *Energies*. 2025; 18(8):1944. https://doi.org/10.3390/en18081944.

[75]. Miao L, Di L, Zhao J, Liu H, Hu Y, Wei X. Optimal Scheduling of Active Distribution Networks with Hybrid Energy Storage Systems Under Real Road Network Topology. *Processes*. 2025; 13(5):1492. https://doi.org/ 10.3390/pr13051492.

[76]. Duan Y, Gao C, Xu Z, Ren S, Wu D. Multi-Objective Optimization for the Low-Carbon Operation of Integrated Energy Systems Based on an Improved Genetic Algorithm. Energies. 2025; 18(9):2283. https://doi.org/10.3390/ en18092283.

[77]. Hu G, Zhao X, Cheng S, Zhang Q, Zhang Q, Bai J, Shi L. The Optimal Configuration of Energy Storage Capacity Based on the NSGA-II Algorithm

and Electrochemical Energy Storage Operational Modes. *Processes*. 2025; 13(5):1432. https://doi.org/10.3390/pr13051432.

# Anexa 1

## Controler fuzzy de temperatură

Reglarea temperaturii spațiului de locuit la consumatorii casnici de energie electrică se face în prezent, în funcție de temperatura interioară (din spațiu de locuit), fără să se aibă în vedere temperatura experioară (din afara spațiului, care depinde de factori climatici și geografici).

Pentru creșterea eficienței energetice a procesului de încălzire prin utilizarea eficientă a surselor de energie, o soluție este adaptarea temperaturii agentului termic (din instalația de încălzire) în funcție atât de termepratura interioară, cât și de cea exterioară. Această soluție se poate implementa prin folosirea unui controler (regulator) fuzzy de temperatură.

Exemplul numeric prezentat aici stă la baza unei lucrări de laborator, care este inclusă în cartea Aplicații ale Inteligenței Artificiale în Managementul Energiei. Suport pentru laborator.

### Controler fuzzy

Regulator fuzzy de temperatură are ca date de intrare temperatura interioară din încăpere și temperatura exterioară. Controlerul va furniza la ieșire tensiunea de alimentare a rezistenței folosită pentru a încălzi agentul termic (care curge prin calorifere). Se consideră că încălzirea se face prin intermediul unei instalații care cuprinde un dispozitiv electrotermice dedicate, care încălzește agentul termic, și un system de țevi și calorifere.

Dezvoltarea controlerului fuzzy presupune parcurgerea a șase etape:

#### *A1.1. Determinarea datelor de intrare și de ieșire*

Alegerea datelor de intrare și ieșire, respective definirea caracteristicilor acestora se realizează după analiza procesului energetic care este controlat, și determinarea variabilelor care influențează procesul energetic și soluțiile posibile pentru creșterea eficiența acestuia.

Pentru definirea caracteristicilor datelor de intrare și ieșire este necesar să existe informații privind modul de variație a acestora.

Se consideră că datele de intrare ale controlerului de temperatură sunt Tin și iText, iar ieșirea este notată cu U.

### *A1.2. Fuzzificarea datelor de intrare și de ieșire*

Fuzzificarea este etapa prin care datele de intrare și ieșire, care inițial au valori singulare, sunt transformate în date fuzzy prin utilizarea unor variabile lingvistice și a unor numere fuzzy atașate acestora. Numere fuzzy care sunt folosite în cadrul laboratorului sunt numere fuzzy triunghiulare, trapezoidale, respectiv rampă / pantă.

**Tin – temperatura interioară**

Tin = [8, 24] = {rece [8, 12], bine [10, 20], cald [18, 24]}

Reprezentarea grafică a celor trei variabile lingvistice, respectiv a numerelor fuzzy corespunzătoare este:

$$\mu_{Tin-rece} = \begin{cases} 1, \ 8 \le Tin < 10 \\ \frac{12-Tin}{12-10}, 10 \le Tin \le 12 \\ 0, Tin > 12 \end{cases} \quad \text{(A1.1)}$$

$$\mu_{Tin-bine} = \begin{cases} \frac{Tin-10}{15-10}, 10 \le Tin \le 15 \\ \frac{20-Tin}{20-15}, 15 \le Tin \le 20 \\ 0, Tin > 20, Tin < 10 \end{cases} \quad \text{(A1.2)}$$

$$\mu_{Tin-cald} = \begin{cases} 1, \ 20 \le Tin < 24 \\ \frac{Tin-18}{20-18}, 18 \le Tin \le 20 \\ 0, Tin < 18 \end{cases} \quad \text{(A1.3)}$$

Reprezentarea grafică a numerelor fuzzy este ilustrată în fig. A1.1.

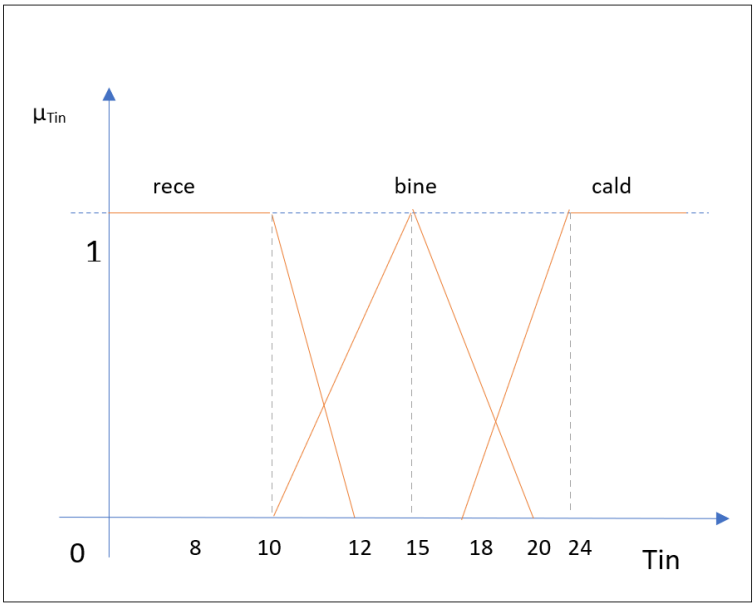



**Fig. A1.1.** Reprezentare grafică Tin

**Text – temperatura exterioară**

Text = [-20, 21]
Text = {rece [-20, 0], bine [0, 15], cald [10, 21]}

Reprezentarea grafică a celor trei variabile lingvistice, respectiv a numerelor fuzzy corespunzătoare este:

$$\mu_{Text-rece} = \begin{cases} 1, \ -20 \leq Text < -10 \\ \frac{0-Text}{0-(-10)}, -10 \leq Text \leq 0 \\ 0, Text > 0 \end{cases} \quad \text{(A1.4)}$$

$$\mu_{Text-bine} = \begin{cases} \frac{Text-(-5)}{0-(-5)}, -5 \leq Text \leq 0 \\ 1, \ 0 < Text \leq 10 \\ \frac{15-Text}{15-10}, 10 < Text \leq 15 \\ 0, Text > 15, Text < -5 \end{cases} \quad \text{(A1.5)}$$

$$\mu_{Text-cald} = \begin{cases} 1, & 20 \leq Text < 21 \\ \frac{Text-10}{20-10}, & 10 \leq Text \leq 20 \\ 0, & Text < 10 \end{cases} \quad \text{(A1.6)}$$

Reprezentarea grafică a numerelor fuzzy este ilustrată în fig. A1.2.

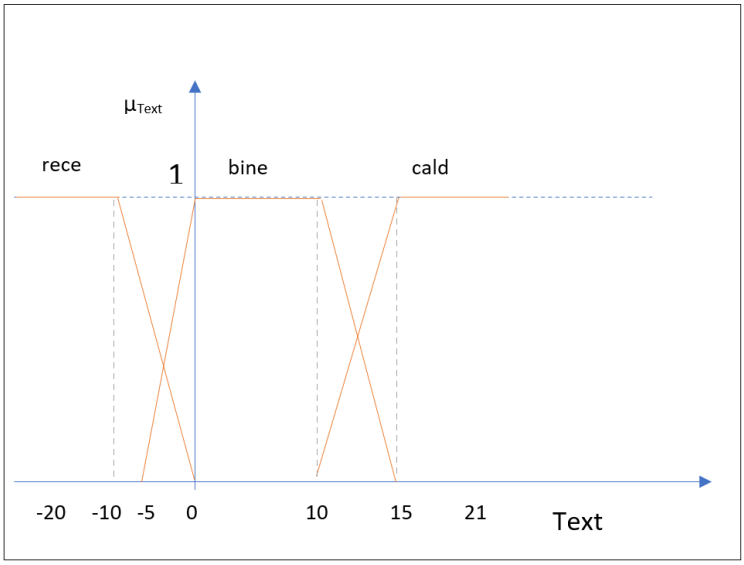



**Fig. A1.2.** Reprezentare grafică Text

**U – tensiunea**

U = [180, 250] = {min [180, 210], med [205, 235], max [225, 250]}
Reprezentarea grafică a celor trei variabile lingvistice, respectiv a numerelor fuzzy corespunzătoare este:

$$\mu_{U-min} = \begin{cases} \frac{U-180}{195-180}, & 180 \leq U \leq 195 \\ \frac{210-U}{210-195}, & 195 < U \leq 210 \\ 0, & U > 210, U < 180 \end{cases} \quad \text{(A1.7)}$$

$$\mu_{U-med} = \begin{cases} \frac{U-205}{220-205}, & 205 \leq U \leq 220 \\ \frac{235-U}{235-220}, & 220 < U \leq 235 \\ 0, & U > 235, U < 205 \end{cases} \quad \text{(A1.8)}$$

$$\mu_{U-max} = \begin{cases} \frac{U-235}{235-240}, 235 \leq U \leq 240 \\ \frac{250-U}{250-240}, 240 < U \leq 250 \\ 0, U > 250, U < 235 \end{cases} \quad \text{(A1.9)}$$

Reprezentarea grafică a numerelor fuzzy este ilustrată în fig. A1.3.

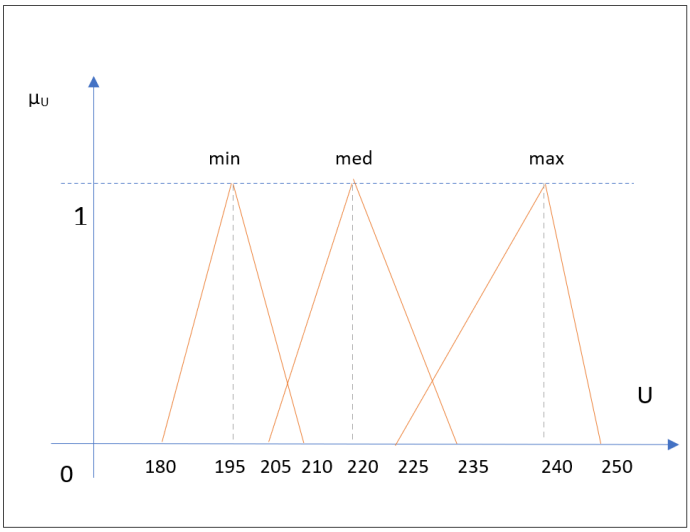



**Fig. A1.3.** Reprezentare grafică U

### *A1.3. Definierea tabelului de inferențe / a regulilor de inferență*

Tabelul de inferențe definește relațiile dintre datele de intrare și ieșire, și reprezintă o formă compactă de a reprezenta regulile de inferență dintre date. În exemplu din lucrare, o regulă de inferență are forma:

Dacă Tin=rece ȘI Text=bine, Atunci U=med.

Tabelul de inferențe este detaliat în continuare, în tabelul A1.1.

**Tabelul A1.1.** Tabelul de inferențe a controlerului fuzzy de temperatură

| U | **Text** | | |
|---|---|---|---|
| | **rece** | **bine** | **cald** |

| | | | | |
|---|---|---|---|---|
| **Tin** | **rece** | max | max | med |
| | **bine** | max | med | min |
| | **cald** | med | min | min |

### *A1.4. Alegerea metodei de rezolvare a inferențelor*

Metodele de rezolvare a inferențelor prezentate în cadrul laboratorului sunt metoda Max-Min, Max-Prod și Sum-Prod. În continuare este exemplificată numeric fiecare metodă de rezolvare a inferențelor.

**Metoda Max-Min**

În folosirea metodei Max-Min, operatorii logici din interiorul regulilor au următoarele semnificații:

ȘI – minim
SAU – maxim
Atunci – minim
SAU (dintre reguli) - maxim

Exemplu

Reguli selectate

1. Dacă Tin=rece (0,2) ȘI Text=rece(0,3), Atunci U=max
2. Dacă Tin=bine (0,3) ȘI Text=bine(0,4), Atunci U=med
3. Dacă Tin=cald (0,2) ȘI Text=cald(0,8), Atunci U=min
4. Dacă Tin=rece (0,5) ȘI Text=rece(0,5), Atunci U=max

Rezolvarea inferențelor

1. Minim (0,2 ; 0,3) => 0,2 => minim (U=max ; 0,2) => număr fuzzy trapezoidal cu înălțimea 0,2
2. Minim (0,3 ; 0,4) => 0,3 => minim (U=med ; 0,3) => număr fuzzy trapezoidal cu înălțimea 0,3
3. Minim (0,2 ; 0,8) => 0,2 => minim (U=min ; 0,2) => număr fuzzy trapezoidal cu înlălțimea 0,2
4. Minim (0,5 ; 0,5) => 0,5 => minim (U=max ; 0,5) => număr fuzzy trapezoidal cu înlălțimea 0,5
5. Maxim ($Umax^{0,2}$, $Umed^{0,3}$, $Umin^{0,2}$, $Umax^{0,5}$) => suprafață fuzzy de ieșire formată din ($Umed^{0,3}$, $Umin^{0,2}$, $Umax^{0,5}$)

**Metoda Max-Prod**

În folosirea metodei Max-Prod, operatorii logici din interiorul regulilor au următoarele semnificații:

ȘI – produs
SAU – maxim
Atunci – produs
SAU (dintre reguli) - maxim

Exemplu

Reguli selectate

1. Dacă Tin=rece (0,2) ȘI Text=rece(0,3), Atunci U=max
2. Dacă Tin=bine (0,3) ȘI Text=bine(0,4), Atunci U=med
3. Dacă Tin=cald (0,2) ȘI Text=cald(0,8), Atunci U=min
4. Dacă Tin=rece (0,5) ȘI Text=rece(0,5), Atunci U=max

Rezolvarea inferențelor

1. Produs (0,2 ; 0,3) => 0,06 => Produs (U=max ; 0,06) => număr fuzzy triunghiular cu înălțimea 0,2
2. Produs (0,3 ; 0,4) => 0,12 => produs (U=med ; 0,12) => număr fuzzy triunghiular cu înălțimea 0,12
3. Produs (0,2 ; 0,8) => 0,16 => produs (U=min ; 0,16) => număr fuzzy triunghiular cu înlălțimea 0,16
4. Produs (0,5 ; 0,5) => 0,25 => produs (U=max ; 0,25) => număr fuzzy triunghiular cu înlălțimea 0,25
5. Maxim ($Umax^{0,06}$, $Umed^{0,12}$, $Umin^{0,16}$, $Umax^{0,25}$) => suprafață fuzzy de ieșire formată din ($Umed^{0,12}$, $Umin^{0,16}$, $Umax^{0,25}$)

**Metoda Sum-Prod**

În folosirea metodei Sum-Prod, operatorii logici din interiorul regulilor au următoarele semnificații:

ȘI – produs
SAU – sumă
Atunci – produs
SAU (dintre reguli) – sumă

Exemplu

Reguli selectate

1. Dacă Tin=rece (0,2) ȘI Text=rece(0,3), Atunci U=max
2. Dacă Tin=bine (0,3) ȘI Text=bine(0,4), Atunci U=med
3. Dacă Tin=cald (0,2) ȘI Text=cald(0,8), Atunci U=min
4. Dacă Tin=rece (0,5) ȘI Text=rece(0,5), Atunci U=max

Rezolvarea inferențelor

1. Produs (0,2 ; 0,3) => 0,06 => Produs (U=max ; 0,06) => număr fuzzy triunghiular cu înălțimea 0,2
2. Produs (0,3 ; 0,4) => 0,12 => produs (U=med ; 0,12) => număr fuzzy triunghiular cu înălțimea 0,12
3. Produs (0,2 ; 0,8) => 0,16 => produs (U=min ; 0,16) => număr fuzzy triunghiular cu înlălțimea 0,16
4. Produs (0,5 ; 0,5) => 0,25 => produs (U=max ; 0,25) => număr fuzzy triunghiular cu înlălțimea 0,25
5. Suma ($Umax^{0,06}$, $Umed^{0,12}$, $Umin^{0,16}$, $Umax^{0,25}$) => suprafață fuzzy de ieșire formată din ($Umed^{0,12}$, $Umin^{0,16}$, $Umax^{0,25+0,06}$)

### *A1.5. Alegerea metodei de defuzzyficare a suprafeței fuzzy de ieșire – determinarea valorii singulare de ieșire .*

Pentru defuzzyficare, în cadrul laboratorului sunt exemplificate metoda centrului de greutate, a centrului sumelor, respectiv a înălțimii. Relațiile matematice corespunzătoare fiecărei metode sunt enumerate în continuare.

#### Metoda centrul de greutate

Prin folosirea acestei metode, se determină valoarea singulară de ieșire ca fiind centrul de greutate a suprafeței fuzzy de ieșire. Dificultatea acestei metode constă în determinarea funcției de apartenență a suprafeței totale de ieșire.

$$U = \frac{A}{B} \tag{A1.10}$$

Unde

$$A = \sum \int u \cdot \mu(u) du \tag{A1.11}$$

$$B = \sum \int \mu(u)du \quad \text{(A1.12)}$$

U – valoarea singulară de ieșire, u – valoare variabilă pe axa 0x, μ(u) – funcția de apartenență a suprafeței fuzzy

**Metoda centrul sumelor**

Prin folosirea acestei metode, se determină valoarea singulară de ieșire ca fiind centrul de greutate a fiecărei suprafețe fuzzy și apoi se determină suma algebrică. Comparativ cu metoda precedentă, în cazul acestei metode nu este necesar să se determine funcția suprafeței totale, ci se vor folosi funcțiile de apartenență a suprafețelor componente.

$$U = \frac{A}{B} \quad \text{(A1.13)}$$

Unde

$$A = \sum \int u \sum \mu(u)du \quad \text{(A1.14)}$$

$$B = \sum \int \sum \mu(u)du \quad \text{(A1.15)}$$

**Metoda înălțimii**

În cazul utilizării metodei de defuzzyficare a suprafeței de ieșire prin folosirea metodei înălțimii se vor folosi valorile înălțimilor suprafețelor componente ale suprafeței totale, și proiecțile acestora pe axa 0x (valorile de pe axa 0x corespunzătoare acestora)

$$U = \frac{A}{B} \quad \text{(A1.16)}$$

Unde

$$A = \sum u \cdot H \quad \text{(A1.17)}$$

$$B = \sum H \quad \text{(A1.18)}$$

H – înălțimea unei suprafețe componente, u – proiecția înălțimii pe axa 0x.

Se presupune suprafața fuzzy din fig. A1.4. Prin aplicarea celor trei metode de defuzzyficare vom avea soluțiile:

Centrul de greutate – relația (A1.19)

$$U = \frac{\int_{180}^{195} u \cdot 0{,}5\frac{u-180}{5}du + \int_{195}^{208} u \cdot \left(0{,}44\frac{208-u}{13} + 0{,}06\right)du + \int_{208}^{220} u \cdot \left(0{,}44\frac{u-208}{12} - 0{,}14\right)du + \int_{220}^{228} u \cdot \left(0{,}44\frac{228-u}{8} - 0{,}14\right)du + \int_{228}^{240} u \cdot (0{,}64\frac{u-228}{12} + 0{,}06)du + \int_{240}^{250} u \cdot 0{,}7\frac{250-u}{10}du}{\int_{180}^{195} 0{,}5\frac{u-180}{5}du + \int_{195}^{208}\left(0{,}44\frac{208-u}{13} + 0{,}06\right)du + \int_{208}^{220}\left(0{,}44\frac{u-208}{12} - 0{,}14\right)du + \int_{220}^{228}\left(0{,}44\frac{228-u}{8} - 0{,}14\right)du + \int_{228}^{240} u \cdot \left(0{,}64\frac{u-228}{12} + 0{,}06\right)du + \int_{240}^{250} 0{,}7\frac{250-u}{10}du} \cong$$

$$223\ V \qquad \text{(A1.19)}$$

Centrul sumelor – relația (A1.20)

$$U = \frac{\int_{180}^{195} u \cdot 0{,}5\frac{u-180}{5}du + \int_{195}^{210} u \cdot \left(0{,}5\frac{210-u}{15} + 0{,}3\frac{u-205}{15}\right)du + \int_{205}^{220} u \cdot \left(0{,}3\frac{u-205}{15} + 0{,}5\frac{210-u}{15}\right)du + \int_{220}^{235} u \cdot \left(0{,}3\frac{235-u}{15} + 0{,}7\frac{u-225}{15}\right)du + \int_{225}^{240} u \cdot (0{,}3\frac{235-u}{15} + 0{,}7\frac{u-225}{15})du + \int_{240}^{250} u \cdot 0{,}7\frac{250-u}{10}du}{\int_{180}^{195} 0{,}5\frac{u-180}{5}du + \int_{195}^{210}\left(0{,}5\frac{210-u}{15} + 0{,}3\frac{u-205}{15}\right)du + \int_{205}^{220}\left(0{,}3\frac{u-205}{15} + 0{,}5\frac{210-u}{15}\right)du + \int_{220}^{235}\left(0{,}3\frac{235-u}{15} + 0{,}7\frac{u-225}{15}\right)du + \int_{225}^{240}(0{,}3\frac{235-u}{15} + 0{,}7\frac{u-225}{15})du + \int_{240}^{250} 0{,}7\frac{250-u}{10}du} \cong$$

$$222\ V \qquad \text{(A1.20)}$$

Metoda înălțimii – relația (A1.21)

$$U = \frac{0{,}5*195+0{,}3*220+0{,}7*240}{0{,}7+0{,}5+0{,}3} = 221\ V \qquad \text{(A1.21)}$$

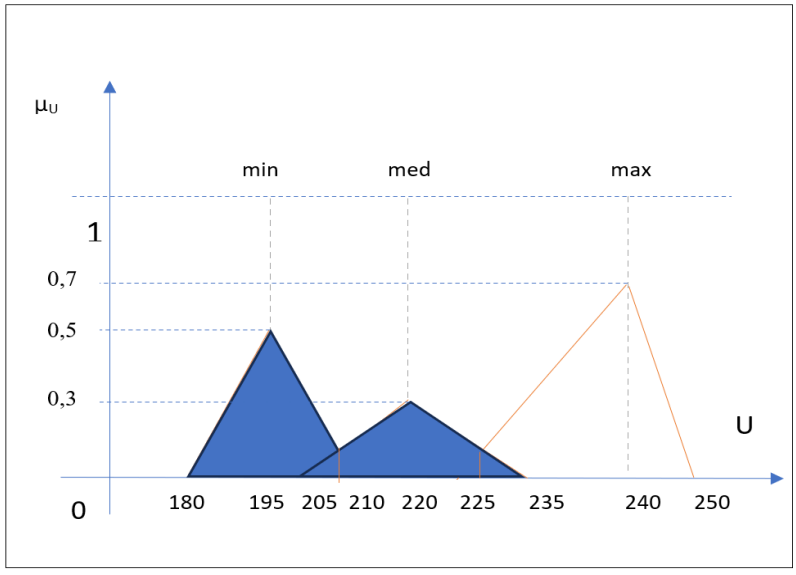



**Fig. A1.4.** Reprezentare grafică a suprafeței de ieșire

### *A1.6. Testarea regulatorului.*

Temperatura interioară se dorește a se menține la valoarea de 21°C.
Se consideră valorile efective ale datelor de intrare:
Tin = 19 °C, Text = -3 °C.

Valorile funcțiilor de apartenență a celor două date de intrare sunt:
Tin-bine = 0,2, Tin-cald = 0,5
Text-rece = 0,3, Text-bine = 0,4

Regulile corespunzătoare datelor de intrare sunt:
1. Dacă Tin=bine ȘI Text=rece, Atunci U=max
2. Dacă Tin=cald ȘI Text=rece, Atunci U=med
3. Dacă Tin=bine ȘI Text=bine, Atunci U=med
4. Dacă Tin=cald ȘI Text=bine, Atunci U=min

Rezolvarea inferențelor din regulile selectate
Metoda Max-Min
Maxim ($Umax^{0,2}$, $Umed^{0,3}$, $Umed^{0,2}$, $Umin^{0,4}$) => suprafață fuzzy (formată din numere fuzzy trapezoidale) de ieșire formată din ($Umed^{0,3}$, $Umin^{0,4}$, $Umax^{0,2}$)

Metoda Max-Prod
Maxim ($Umax^{0,2}$, $Umed^{0,3}$, $Umed^{0,2}$, $Umin^{0,4}$) => suprafață fuzzy (formată din numere fuzzy triunghiulare) de ieșire formată din ($Umed^{0,3}$, $Umin^{0,4}$, $Umax^{0,2}$)

Metoda Sum-Prod
Suma ($Umax^{0,06}$, $Umed^{0,15}$, $Umin^{0,2}$, $Umed^{0,08}$) => suprafață fuzzy (formată din numere fuzzy triunghiulare) de ieșire formată din ($Umed^{0,23}$, $Umin^{0,2}$, $Umax^{0,06}$)

# Anexa 2

# Rețea neuronală artificială pentru prognoza pe termen scurt a consumului de energie electrică

Pentru creșterea calității activității de management energetic a unui consumator comercial, o soluție este estimarea consumului de energie electrică pe ziua următoare, adică prognoza pe termen scurt.

În acest scop, se va folosi o rețea neuronală artificială de tipul multistrat Perceptron cu un singur strat ascuns. Neuroni artificiali ai RNA sunt modelați după modelul neuronului static, care este activat prin funcția sigmoid.

Antrenarea (învățarea) rețelei se realizează prin implicarea metodei de învățarea supravegheată și propagarea înapoi a erorii (error back-propagation).

Exemplul numeric prezentat stă la baza unei lucrări de laborator inclusă în materialul didactic corespunzător, și anume Aplicații ale Inteligenței Artificiale în Managementul Energiei. Suport pentru laborator.

## Rețeaua Neuronală Artificială

### *A2.1. Arhitectura RNA*

Rețeaua neuronală artificială este formată din trei straturi de neuroni, figura A2.1. Primul strat, stratul de intrare cuprinde cinci neuroni de intrare, al doilea strat, stratul ascuns are trei neuroni, iar ultimul strat, stratul de ieșire are un singur neuron.

Stratul de intrare are rolul de a prelua datele de intrare și a le transmite mai departe, către al doilea strat, cel ascuns. Cei 5 neuroni de intrare, $x_1$....$x_5$ corespund factorilor care influențează consumul de energie electrică, și anume: temperatura mediului ambiant, umiditatea mediului ambiant, nivelul de iluminare naturală, tipul de zi și numărul de angajați. Valoarea neuronilor de intrare se modifică în funcție de valorile datelor de intrare, ci nu prin calcule în cadru procesului de antrenare. Valorile datelor de intrare sunt preluate de la exterior, de cele mai multe ori fiind scalate pentru a ușura calculele, altfel mărimile lor sunt neschimbate.

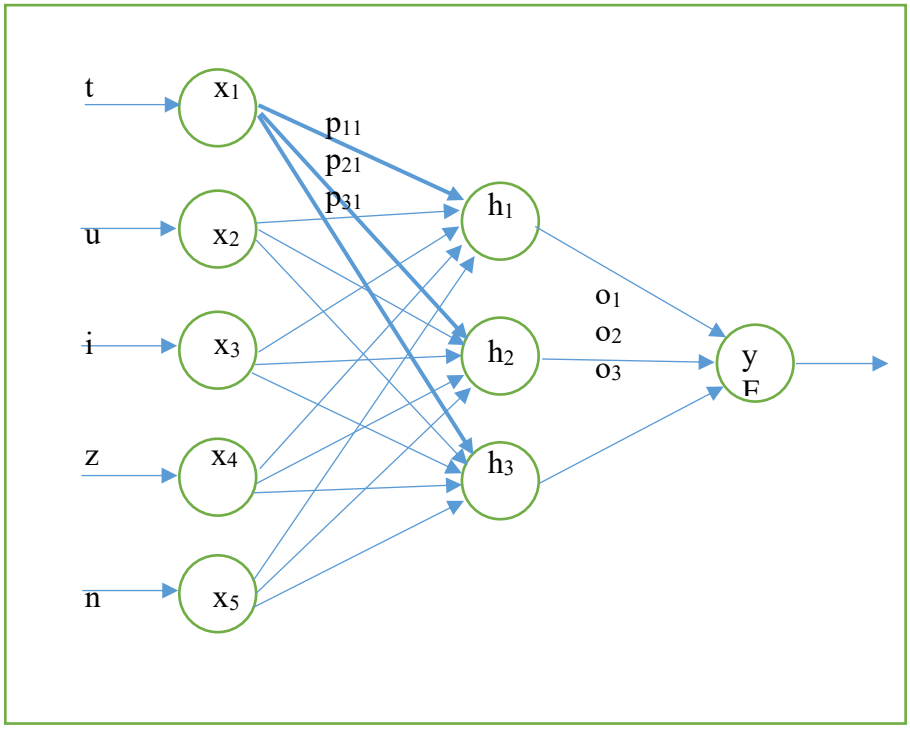



**Fig. A2.1.** Arhitectura RNA

Temperatura mediului ambiant, t [°C] are intervalul de variație [-20, 35], este un factor care influențează consumul de energie electrică prin intermediul receptoarelor necesare pentru încălzire în timpul iernii și răcire în timpul verii, respectiv răcirea produselor perisabile pe tot parcursul anului.

Umiditatea mediului ambiant, u [%] variază în intervalul [40, 80]. Acest factor influențează consumul de energie electrică prin faptul că afectează procesul de răcire și încălzire.

Nivelul de iluminare naturală, i [lx] poate varia în intervalul [200, 1000], el afectând consumul de energie electrică a consumatorului prin intermediul aparatelor de iluminat.

Tipul zilei este un factor notat, cu n [1, 2, ..., 10], care a fost introdus deoarece studiile de specialitate au evidențiat faptul că forma curbei de sarcină a consumatorilor comerciali este diferită în funcție de ziua săptămânii (zi de lucru, zi de weekend) și de perioade speciale ale anului (concedii, sărbători).

Numărul de angajați, notat cu z, care ia valori numere naturale între 1 și 10, este un factor care influențează consumul de energiei electrică a consumatorului,

întrucât numărul de angajați prezenți infuențează numărul de receptoare care sunt folosite.

Stratul ascuns este compus din trei neuroni, notați cu $h_1$, $h_2$ și $h_3$. Rolul acestor neuroni este de a înmagazina informațiile intermediare în etapele de antrenare, testare și folosire a RNA. Diferit de neuronii din stratul de intrare, valorile neuronilor de stratul ascuns se modifică permanent.

Stratul de ieșire a RNA conține un singur neuron, y, care corespunde consumului de energie electrică (în valori scalate), E [kWh] și va indica valoarea acesteia în intervalul [2, 10].

*Observație*

Pentru a dezvolta o aplicație RNA este necesară o bază consistentă de date. Aceste date vor fi folosite pentru a identifica factorii care influențează prognoza, intervalele de variație a acestora, precum și a ieșirii, dar în primul rând pentru a antrena și testa rețeaua.

Datele care sunt prezentate în acest exemplu sunt în scop didactic, deci nu caracterizează în mod fidel consumul de energie electrică a unui consumator comercial.

Precum se observă în fig. A2.1, straturile rețelei au neuroni total interconectați, astfel toți neuroni din stratul de intrare sunt conectați cu toți neuronii din stratul ascuns, iar aceștia din urmă sunt conectați cu neuronul de ieșire. Deci, rețeaua este total interconectată prin intermediul unor săgeți unidirecționale, denumite $p_{ij}$ (i este indicele neuronului sursă și j este indicele neuronului receptor), respectiv $o_1$, $o_2$ și $o_3$. Săgețile indică faptul că informația va pleca de la neuroni din straturile inferioare, înspre neuroni din straturile superioare. Aceste conexiuni dintre neuroni se numesc ponderile neuronilor (weights) și indică modul în care o informație care trece de la un neuron dintr-un strat inferior înspre unul superior are importanță (influențează) în determinarea valorii finale a neuronului receptor. Ponderile au valori pozitive subunitare, adică aparțin intervalului [0, 1].

Datele de intrare și ieșire, precum și ponderile sunt reprezentate sub forma de matrici uni sau bidimensionale. Relațiile (A2.1) – (A2.5) descriu forma de prezentare a neuronilor din cele trei straturi și a ponderilor dintre neuroni când RNA este implementată matematic și informatic.

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \quad \text{(A2.1)}$$

$$H = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} \quad \text{(A2.2)}$$

$$Y = [y] \quad \text{(A2.3)}$$

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \\ p_{41} & p_{42} & p_{43} \\ p_{51} & p_{52} & p_{55} \end{bmatrix} \quad \text{(A2.4)}$$

$$O = \begin{bmatrix} o_1 \\ o_2 \\ o_3 \end{bmatrix} \quad \text{(A2.5)}$$

### *A2.2. Datele de antrenare și de testare*

Datele de antrenare și testare a RNA sunt cei mai importanți factori care influențează eficiența RNA. Astfel reiese atenția care trebuie acordată acestora.

În acest exemplu au fost considerate cinci seturi de date, trei pentru antrenare și două pentru testare. Un set conține valorile datelor de intrare și ieșirea corespunzătoare. Tabelul A2.1 prezintă seturile de antrenament și testare.

**Tabelul A2.1**. Datele de antrenament și testare ale RNA

| **Set** | **t** | **u** | **i** | **z** | **n** | **E** |
|---|---|---|---|---|---|---|
| 1 | 20 | 60 | 1000 | 1 | 5 | 6 |
| 2 | -20 | 40 | 200 | 1 | 5 | 8 |
| 3 | 35 | 80 | 500 | 2 | 5 | 10 |
| 4 | 10 | 40 | 800 | 10 | 10 | 5 |
| 5 | 5 | 70 | 500 | 7 | 1 | 2 |

Din aceste seturi de antrenament se vor folosi setul 1, 3 și 5 pentru antrenarea rețelei și setul 2 și 4 pentru testarea ei.

Având în vedere că rețeaua lucrează cu numere subunitare în modul, datele trebuie scalate pentru a corespunde acestei cerințe. Metoda cea mai folosită în literatura de scpecialitate este scalarea după valoarea maximă a unui parametru. Însă, în acest exemplu se va folosi scalarea cu multimpli de 10. Astfel, valorile lui t și u vor fi împărțite cu 100, ale lui z, n și E cu 10, iar ale lui i cu 1000. În consecință, noul set de date care va fi folosit mai departe de rețea este cel din tabelul A2.2.

**Tabelul A2.2.** Datele de antrenament și testare ale RNA scalate

| **Set** | **t** | **u** | **i** | **z** | **n** | **E** |
|---|---|---|---|---|---|---|
| 1 | 0,2 | 0,6 | 1 | 0,1 | 0,5 | 0,6 |
| 2 | -0,2 | 0,4 | 0,2 | 0,1 | 0,5 | 0,8 |
| 3 | 0,35 | 0,8 | 0,5 | 0,2 | 0,5 | 1 |
| 4 | 0,1 | 0,4 | 0,8 | 1 | 1 | 0,5 |
| 5 | 0,05 | 0,7 | 0,5 | 0,7 | 0,1 | 0,2 |

Seturile de antrenament și testare scriese sub forma unor matrici, care sunt notate cu A (setul de antrenament) și T (setul de testare) au valorile din relațiile (A2.6) și (A2.7).

$$A = \begin{bmatrix} 0,2 & 0,6 & 1 & 0,1 & 0,5 & 0,6 \\ 0,35 & 0,8 & 0,5 & 0,2 & 0,5 & 1 \\ 0,05 & 0,7 & 0,5 & 0,7 & 0,1 & 0,2 \end{bmatrix} \quad \text{(A2.6)}$$

$$T = \begin{bmatrix} -0,2 & 0,4 & 0,2 & 0,1 & 0,5 & 0,8 \\ 0,1 & 0,4 & 0,8 & 1 & 1 & 0,5 \end{bmatrix} \quad \text{(A2.7)}$$

### *A2.3. Antrenarea RNA*

Antrenarea supravegheată a RNA este procesul prin care rețeaua ”învață” să asocieze datele de intrare la ieșirea corespunzătoare. Această învățare se realizează prin modificarea și adaptarea continuă în timpul antrenării a ponderilor dintre neuroni.

În prima etapă a antrenării sunt date valori aleatorii ponderilor în intervalul 0, 1. Aceste valori apoi prin relațiile matermatice corespunzătoare (propagarea înapoi a erorii) se vor modifica, într-un final, la sfârșitul procesului de antrenare

vor avea niște valori care rămân fixe în procesul de testare și urilizare ulterioară a RNA.

Valorile ponderilor P și O în proma fază a antrenării sunt cele din relațiile (A2.8) și (A2.9).

$$P = \begin{bmatrix} 0,1 & 0,2 & 0,3 \\ 0,2 & 0,3 & 0,4 \\ 0,1 & 0,2 & 0,3 \\ 0,2 & 0,3 & 0,4 \\ 0,1 & 0,2 & 0,3 \end{bmatrix} \quad \text{(A2.8)}$$

$$O = \begin{bmatrix} 0,1 \\ 0,2 \\ 0,3 \end{bmatrix} \quad \text{(A2.9)}$$

Relațiile matematice necesare pentru a determina activarea neuronilor, respectiv porpagarea înapoi a erorii sunt (A2.10) – (A2.18).

Relația (A2.10) se folosește pentru a determina valoarea corespunzătoare corpului unui neuron static, iar relația (A2.12) pentru a calcula activarea lui, adică valoarea care se transmite mai departe, la neuronii din stratul superior.

$$net_{h_i} = \sum x_j \cdot p_{ji} \quad \text{(A2.10)}$$

unde i = 1, 2, 3, iar j = 1,.., 5

iar pentru determinarea valorii lui $h_1$

$$net_{h_1} = x_1 \cdot p_{11} + x_2 \cdot p_{21} + x_3 \cdot p_{31} + x_4 \cdot p_{41} + x_5 \cdot p_{51} \quad \text{(A2.11)}$$

$$h_i = \frac{1}{1+e^{-net_{hi}}} \quad \text{(A2.12)}$$

unde i = 1, 2, 3.

Valoarea corpului neuronului de ieșire se determină cu relațiile (A2.13) și (A2.14), iar valoarea de ieșire, notată cu yc, deoarece este obținută prin calcule se obține din relația (A2.15).

$$net_{yc} = \sum h_i \cdot o_i \quad \text{(A2.13)}$$

unde i = 1, 2, 3,

iar forma desfășurată este

$$net_{yc} = h_1 \cdot o_1 + h_2 \cdot o_2 + h_3 \cdot o_3 \quad \text{(A2.14)}$$

$$y_c = \frac{1}{1+e^{-net_{yc}}} \quad \text{(A2.15)}$$

Prin relațiile prezentate anterior, adică (A2.10) – (A2.15), se realizează transmiterea informațiilor de la stratul de intrare înspre stratul de ieșire. În continuare, ieșirea calculată este comparată cu ieșirea din setul de antrenament, iar diferența obținută (dacă este diferită de zero, sau o limită foarte mică) va fi considerată eroarea, care se propagă înapoi pentru a modifica valorile ponderilor. După adaptarea ponderilor, acestea vor corespunde setului de antrenament folosit.

$$err = y_D - y_c \quad \text{(A2.16)}$$

$$o_i^* = o_i + \Delta o_i \quad \text{(A2.17)}$$

unde noua valoare adaptată a ponderii este $o_i^*$, $\alpha$ este rata de învățare, care ia valori în intervalul [0, 1], $\Delta o_i = \alpha \cdot h_i \cdot \delta o_i$, $\delta o_i = h_i(1 - h_i) \cdot err$. $\Delta o_i$ este valoarea cu care se modifică ponderea, iar $\delta o_i$ este ponderea din eroare care revine lui $o_i$, iar *err* este eroarea mărimii calculate față de cea dorită, care se regăzește în setul de antrenament, $y_D$ – ieșirea dorită, $y_c$ – ieșirea calculată.

$$p_{ij}^* = p_{ij} + \Delta p_{ij} \quad \text{(A2.18)}$$

unde noua valoare adaptată a ponderii este $p_{ij}^*$, $\alpha$ este rata de învățare, care ia valori în intervalul [0, 1], $\Delta p_{ij} = \alpha \cdot x_j \cdot \delta p_{ij}$, $\delta p_{ij} = x_j(1 - x_j) \cdot \delta o_i \cdot o_i$; $\Delta p_{ij}$ este valoarea cu care se modifică ponderea, iar $\delta p_{ij}$ este ponderea din eroare care revine lui $p_{ij}$.

Relațiile matematice (A2.16)-(A2.18) arată raționamentul matematic prin care se propagă înapoi eroarea dintre ieșirea calculată și cea dorită. Cu ajutorul relației matematice (A2.17) se determină noile valori adaptate ale ponderilor care leagă neuronii din stratul ascuns de cel de ieșire. Relația matematică (A2.18) arată modul în care se modifică ponderile dintre neuronii din stratul de intrare și neuronii din stratul ascuns. Se poate observa că noua valoare este influențată de modificările realizate la ponderile din etapa precedentă.

Antrenarea începe prin parcurgerea setului de antrenament o dată, și adaptarea ponderilor, astfel se realizează o epocă. În antrenarea unei RNA este nevoie de mai multe epoci. Aici se va folosi dor o epocă.

Setul 1

Datele cu care lucrează rețeaua la început sunt:

$$X = \begin{bmatrix} 0,2 \\ 0,6 \\ 1 \\ 0,1 \\ 0,5 \end{bmatrix}, H = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}, Y_D = [0,6], P = \begin{bmatrix} 0,1 & 0,2 & 0,3 \\ 0,2 & 0,3 & 0,4 \\ 0,1 & 0,2 & 0,3 \\ 0,2 & 0,3 & 0,4 \\ 0,1 & 0,2 & 0,3 \end{bmatrix}, O = \begin{bmatrix} 0,1 \\ 0,2 \\ 0,3 \end{bmatrix}$$

După realizarea calculelor, se obțin următoarele rezultate:

$$NET_H = \begin{bmatrix} 0,31 \\ 0,55 \\ 0,79 \end{bmatrix}, H = \begin{bmatrix} 0,57 \\ 0,63 \\ 0,68 \end{bmatrix}, \quad Y_D = [0,6], \quad y_D = 0,6, \quad net_{yc} = 0,38, \quad y_c = 0,595 \cong 0,6$$

Întrucât $err = 0$, nu este nevoie ca ponderile să fie adaptate.

Setul 2

Datele cu care lucrează rețeaua la început sunt:

$$X = \begin{bmatrix} 0,35 \\ 0,8 \\ 0,5 \\ 0,2 \\ 0,5 \end{bmatrix}, H = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}, Y_D = [1], P = \begin{bmatrix} 0,1 & 0,2 & 0,3 \\ 0,2 & 0,3 & 0,4 \\ 0,1 & 0,2 & 0,3 \\ 0,2 & 0,3 & 0,4 \\ 0,1 & 0,2 & 0,3 \end{bmatrix}, O = \begin{bmatrix} 0,1 \\ 0,2 \\ 0,3 \end{bmatrix}$$

După realizarea calculelor, se obțin următoarele rezultate:

$$NET_H = \begin{bmatrix} 0,33 \\ 0,57 \\ 0,8 \end{bmatrix}, H = \begin{bmatrix} 0,58 \\ 0,63 \\ 0,69 \end{bmatrix}, Y_D = [1], y_D = 1, net_{yc} = 0,39, y_c = 0,6$$

Întrucât $err = 0,4$ este nevoie ca ponderile să fie adaptate. Astfel, după realizarea calculelor și propagarea înapoi a erorii, noile ponderi au valorile enumerate în continuare.

$$O^* = \begin{bmatrix} 0,1056 \\ 0,2058 \\ 0,3059 \end{bmatrix} \cong \begin{bmatrix} 0,11 \\ 0,21 \\ 0,31 \end{bmatrix},$$

$$P^* = \begin{bmatrix} 0{,}1005 & 0{,}2005 & 0{,}3005 \\ 0{,}2005 & 0{,}3005 & 0{,}4005 \\ 0{,}1005 & 0{,}2005 & 0{,}3005 \\ 0{,}2005 & 0{,}3005 & 0{,}4005 \\ 0{,}1005 & 0{,}2005 & 0{,}3005 \end{bmatrix} \cong \begin{bmatrix} 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \end{bmatrix},$$

Se poate observa din noilor valori ale ponderilor faptul că, ponderile dintre stratul de ieșire și cel ascuns (care sunt mai aproape de eroare) au fost mai puternic afectate, față de ponderile dintre stratul de intrare și cel ascuns.

Setul 3

Datele cu care lucrează rețeaua la început sunt:

$$X = \begin{bmatrix} 0{,}05 \\ 0{,}7 \\ 0{,}5 \\ 0{,}7 \\ 0{,}1 \end{bmatrix}, \quad H = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}, \quad Y_D = [0{,}2], \quad P = \begin{bmatrix} 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \end{bmatrix}, O = \begin{bmatrix} 0{,}11 \\ 0{,}21 \\ 0{,}31 \end{bmatrix}$$

După realizarea calculelor, se obțin următoarele rezultate:

$$NET_H = \begin{bmatrix} 0{,}34 \\ 0{,}55 \\ 0{,}75 \end{bmatrix}, H = \begin{bmatrix} 0{,}58 \\ 0{,}63 \\ 0{,}68 \end{bmatrix}, Y_D = [0{,}2],\ y_D = 0{,}2,\ net_{yc} = 0{,}4,\ y_c = 0{,}6$$

Întrucât $err = -0{,}4$ este nevoie ca ponderile să fie adaptate. Astfel, după realizarea calculelor și propagarea înapoi a erorii, noile ponderi au valorile enumerate în continuare.

$$O^* = \begin{bmatrix} 0{,}1043 \\ 0{,}2041 \\ 0{,}3040 \end{bmatrix} \cong \begin{bmatrix} 0{,}1 \\ 0{,}2 \\ 0{,}3 \end{bmatrix}, \qquad P^* = \begin{bmatrix} 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \end{bmatrix},$$

Se poate observa din noilor valori ale ponderilor faptul că, ponderile O s-au modificat foarte puțin, iar ponderile P foarte puțin. Acest proces de adaptare a ponderilor continuă până la îndeplinirea condiției de oprire. În cazul de față

condiția de oprire este parcurgerea unei epoci, dar în mod uzual, se efectuează peste 50 de epoci pentru ca antrenarea să dea rezultate bune.

După finalizarea antrenării RNA, ponderile rețelei vor fi adaptate pentru a corespunde tuturor datelor de intrare, deci rețeaua va fi capabilă să asocieze datele de intrare la ieșirea corespunzătoare. Aceste ponderi sunt utilizate ulterior în etapa de testare a rețelei.

### *A2.4. Testarea RNA*

În etapa de testare, rețeaua are următoarea formă

$$X = \begin{bmatrix} 0{,}05 \\ 0{,}7 \\ 0{,}5 \\ 0{,}7 \\ 0{,}1 \end{bmatrix} \rightarrow P = \begin{bmatrix} 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \end{bmatrix} \rightarrow H = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} \rightarrow O = \begin{bmatrix} 0{,}1 \\ 0{,}2 \\ 0{,}3 \end{bmatrix} \rightarrow Y = [y]$$

Seturile de testare ale rețelei cu ajutorul cărora se va determina eficacitatea rețelei sunt descrise în relația (7).

$$T = \begin{bmatrix} -0{,}2 & 0{,}4 & 0{,}2 & 0{,}1 & 0{,}5 & 0{,}8 \\ 0{,}1 & 0{,}4 & 0{,}8 & 1 & 1 & 0{,}5 \end{bmatrix}$$

Aceste valori de testare for fi folosite pentru a transmite informația dinspre stratul de start, înspre stratul de ieșire. Rezultatul obținut este comparat cu datele de ieșire din seturile de testare. Dacă rezultatele au o deviație mai mică de 1% (datele calculate au o deviație față de cele dorite) în peste 90% din cazurile (seturi) testate.

Setul 1

Datele cu care lucrează rețeaua la început sunt:

$$X = \begin{bmatrix} -0{,}2 \\ 0{,}4 \\ 0{,}2 \\ 0{,}1 \\ 0{,}5 \end{bmatrix}, \quad H = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}, \quad Y_D = [0{,}8], \quad P = \begin{bmatrix} 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \end{bmatrix}, O = \begin{bmatrix} 0{,}1 \\ 0{,}2 \\ 0{,}3 \end{bmatrix}$$

După realizarea calculelor, se obțin următoarele rezultate:

$$NET_H = \begin{bmatrix} 0{,}15 \\ 0{,}25 \\ 0{,}35 \end{bmatrix}, H = \begin{bmatrix} 0{,}53 \\ 0{,}56 \\ 0{,}59 \end{bmatrix}, Y_D = [0{,}8],\ y_D = 0{,}8,\ y_c = 0{,}58$$

Deviația ieșirii calculate față de cea dorită este de *dev* = 0,22, valoare care depășește procentul de 1%.

Setul 2

Datele cu care lucrează rețeaua la început sunt:

$$X = \begin{bmatrix} 0{,}1 \\ 0{,}4 \\ 0{,}8 \\ 1 \\ 1 \end{bmatrix}, H = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}, Y_D = [0{,}4], P = \begin{bmatrix} 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}4 \\ 0{,}1 & 0{,}2 & 0{,}3 \end{bmatrix}, O = \begin{bmatrix} 0{,}1 \\ 0{,}2 \\ 0{,}3 \end{bmatrix}$$

După realizarea calculelor, se obțin următoarele rezultate:

$$NET_H = \begin{bmatrix} 0{,}47 \\ 0{,}8 \\ 1{,}13 \end{bmatrix}, H = \begin{bmatrix} 0{,}61 \\ 0{,}69 \\ 0{,}76 \end{bmatrix}, Y_D = [0{,}4],\ y_D = 0{,}4,\ y_c = 0{,}63$$

Deviația ieșirii calculate față de cea dorită este de *dev* = -0,22, valoare care depășește procentul de 1%.

Din rezultatele obținute se poate observa că RNA nu a fost antrenată corespunzător, astfel nu a reușit să treacă etapa de testare. În această situație, RNA va fi antrenată din nou pe mai multe epoci, apoi testată din nou, procesul se va repeta până când datele obținute în urma testării corespund cerințelor.

## Anexa 3

# Optimizarea procesului de producție a unui consumator industrial prin utilizarea algortimilor genetici

Un aspect important în managementul energetic este controlul consumului de energie electrică. Implicarea argoritmilor genetici în optimizarea consumului de energie electrică la consumatori este un fapt care se regăsește în multe studii din literatura de specialitate. Pentru a înțelege modul de etapele dezvoltării și modul de operare a unui AG, în continuare este prezentat un exemplu numeric cu scop didactic (care stă la baza unei lucrări de laborator prezentată în materialul didactic corespunzător – Aplicații ale Inteligenței Artificiale în Managementul Energiei. Suport pentru laborator).

Un consumator industrial folosește în procesul tehnologic de producție 12 echipamente dedicate obținerii unor produse. Acestea se caracterizează prin atribute specifice care le clasifică în patru clase de calitate. Echipamentele electrice se împart în mai patru categorii în funcție de calitatea produselor care le produc, puterea activă și numărul de produse care îl produc zilnic.

Producția și funcționarea (echipamentele folosite) consumatorului industrial depinde de client și contractul cu acesta, deoarece, în contract se specifică numărul de produse și calitatea acestora. Pe baza acestor doi parametrii (numărul și calitatea produselor), consumatorul va pune în funcțiune anumite echipamente.

În această situație, obiectivul consumatorului industrial este de a folosi combinația optimă de punere în funcțiune a echipamentelor electrice astfel încât să obțină produsele din contract cu costurile cele mai mici, adică consum minim de energie electrică.

În problemă ne intereseaază doar trei caracteristici ale echipamentelor electrice, și anume puterea activă, $P$ (10 și 200 kW), calitatea produselor $q$, care poate lua valorile 1, 2, 3, 4 (1 – calitatea cea mai bună, și 4 – calitatea cea mai slabă), și numărul de produse obținute într-o zi, $n$, variază între 2 și 10 bucați/zi.

Tabelul A3.1 descrie cele 12 echipamente electrice ale consumatorului industrial cu cele trei atribute ale lor.

**Tabelul A3.1.** Echipamentele consumatorului insutrial.

| Echipament | P [kW] | q | n [bucăți/zi] |
|---|---|---|---|
| 1 | 10 | 1 | 2 |
| 2 | 80 | 1 | 10 |
| 3 | 200 | 1 | 20 |
| 4 | 20 | 2 | 5 |
| 5 | 100 | 2 | 30 |
| 6 | 150 | 2 | 50 |
| 7 | 50 | 3 | 10 |
| 8 | 90 | 3 | 45 |
| 9 | 180 | 3 | 80 |
| 10 | 70 | 4 | 20 |
| 11 | 12 | 4 | 60 |
| 12 | 200 | 4 | 100 |

Pentru înțelegerea mai ușoară a modului de operare a algoritmului genetic, acest exemplu se particularizează pentru un contract specific, care este descris în paragraful următor.

Un client prin intermediul unui contract cere următoarele produse: 50 bucăți porduse calitatea 1, 100 bucăți produse calitatea 2, 150 produse calitatea 3 și 200 produse calitatea 4. În plus, se specifică faptul că produsele de calitate inferioară pot fi înlocuite (în număr limitat) cu produse de calitate superioară.

Se va folosi un algoritm genetic pentru a determina soluția optimă de utilizare a echipamentelor pentru a obține produsele din contract în cel mai scurt timp și utilizând energia electrică în mod eficient. Mai mult, dacă va fi necesar, se va folosi acordul din contract privind înlocuirea produselor de calitate inferioară cu cele de calitate superioară.

### Algoritmul Genetic

Algoritmul genetic lucrează cu soluțiile problemei, adică funcționarea echipamentelor consumatorului industrial. Prima etapă în dezvoltarea algoritmului este codificarea soluțiilor, care se referă la modul în care se vor reprezenta acestea. Următorul pas este determinarea funcției obiectiv și de adaptare, apoi dezvoltarea primei populații, evaluarea indivizilor și construirea noilor generații.

### *A3.1. Codificarea soluției*

Codificarea soluției se va realiza prin folosirea valorilor {0, 1}, adică codificarea binară. Astfel, pentru fiecare echipament se va folosi una din cele două valori, care va indica starea de funcțiune a echipamentului corespunzător. Mai exact, 0 – echipamentul nu funcționează, iar 1 – echipamentul funcționează.

O soluție este reprezentată sub forma unui cromozom cu 12 gene, alele fiind 0 și 1, fig. A3.1. În figură, se observă echipamentele notate E1,...., E12., care corespund genelor cromozomului.

| E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | E10 | E11 | E12 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |

**Fig. A3.1.** Reprezentarea unei soluții / cromozom

Adaptarea unui cromozom (analiza soluției față de soluția optimă) este cuantificată prin folosirea unor criterii clar definite. În plus, față de această funcție de adaptare, este definită funcția obiectiv. De multe ori, funcția de adaptare derivă din funcția obiectiv. În problema descrisă se dorește un consum minim de energiei electrică, dar și obținerea produselor. Aceste funcții sunt prezentate în următorul subcapitol.

### *A3.2. Funcțiile obiectiv și de adaptare*

Obiectivul acestei probleme, a algoritmului genetic este descris prin trei relații matematice și o condiție restrictivă. Astfel, sunt definite trei funcții, Fo1, Fo2 și Fo3 și restricția R, prin (A3.1) – (A3.3).

$$Fo1_j = \sum_{i=1}^{m} c_q * n_i \qquad \text{(A3.1)}$$

Unde Fo1 este prima funcție care definește obiectivul prin care se determină apropierea cromozomului de obiectiv din punctul de vedere a numărului și calitatea propduselor, $j$ reprezintă indicele cromozomul din mulțimea de soluții (populația de cromozomi / indivizi), $i$ este indicele echipamentului / a genei din cromozom, $m$ = 12 este numărul de echipamente, $q$ – calitatea produsului, $c_q$ = 10 pentru $q$=1, $c_q$=7 pentru $q$=2, $c_q$ = 4 pentru $q$=3, și $c_q$ = 1 pentru $q$=4. Se poate observa că la această funcție se urmărește maximizarea ei înspre valoarea maximă posibilă $Fo1_{max}$ = 2000. Această valoare s-a obținut pentru produsele din contract.

$$Fo2_j = \sum_{i=1}^{n} P_i \ [kW] \qquad \text{(A3.2)}$$

Unde *Fo*2 reprezintă a doua funcție a obiectivului prin care se determină puterea activă a echipamentelor în funcțiune ale soluției *j*, $P_i$ este puterea echipamentului *i*.

$$Fo3_j = \frac{F1_{max}}{F2_j} \tag{A3.3}$$

Unde *Fo*3 este timpul în zile necesare soluției *j* să realizeze produsele din contract.

Din relațiile (A3.2) și (A3.3) rezultă consumul de energie a unei soluții/cromozom, care se dorește a se minimiza.

$$Cons_j = Fo2_j \cdot Fo3_j \cdot 24\ [kWh] \tag{A3.4}$$

Pentru a respecta contractul, se impune o condiție restrictivă, și anume funcționarea a cel puțin a unui echipament din fiecare categorie de produs. Dacă un cromozom nu respectă această condiție, atunci nu va fi considerat viabil. Matematic, valoarea lui *R* se determină prin relația următoare.

$$R_j = \begin{cases} 1, & toate\ E_i^q = 0,\ q = \overline{1,4} \\ 0, & contrar. \end{cases} \tag{A3.5}$$

Unde $E_i^q$ este gena *i* a cromozomului *j* corespunzătoare unui echipament care produce produse din categoria $q$. Din relație, rezultă că valoarea restricției poate fi 0 sau 1.

Luând în considerare aceste funcții, se definește funcția de adaptare (fitness function – funcția de potrivire) *F*, care arată cât de apropiată de soluția optimă este un cromozom.

$$F_j = \frac{Fo1_j}{\sum_{j=1}^{m} Fo1_j} \tag{A3.6}$$

Din această relație, rezultă că funcția de adaptare este pozitivă, subunitară, iar valoarea ei se urmărește să se maximizeze. În concluzie, un cromozom cu cât are funcția de adaptare mai mare, cu atât este mai potrivit, și are șanse mari să fie folosit în continuare.

### *A3.3. Prima populație*

Următorul pas în dezvoltarea algoritmului genetic este determinarea numărului de cromozomi / indivizi ai populației, adică numărul de soluții din mulțimea de soluții cu care va lucra algoritmul genetic. Acest număr rămâne

constant pe tot parcursul utilizării algoritmului. În acest exemplu, numărul de indivizi, *m* este 10.

Valorile primei populații sunt generate aleator, conform codificării binare.

Pentru exemplu rezentat s-a considerat ca populația are 10 indivizi. Prima populație este generată aleator și prezentată în fig. A3.2.

Primul cromozom al populației are următoarea semnificație - întrucât echipamentele E1, E3, E5, E7, E9 și E11 au asociat valoarea 1, rezultă că acestea vor fi luate în considerare ca fiind în funcțiune, deci atunci când se calculează valoriile funcțiilor obiectiv, din tabelul 1 se vor extrage caracteristicile acestor echipamente.

Odată determinată prima populație, pentru fiecare dintre cromozomi se determină valorile funcțiilor Fo1, Fo2, Fo3 și R. Rezultalele obținute pentru acești cromozomi sunt ilustrate în tabelul A3.2.

| | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | E10 | E11 | E12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| C2 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| C3 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| C4 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| C5 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| C6 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| C7 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| C8 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| C9 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| C10 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |

**Fig. A3.2.** Prima populație de indivizi. C - cromozom

**Tabelul A3.2.** Valorile funcțiilor obiectiv pentru prima populație

| **Cromozom** | **Fo1** | **Fo2** | **Fo3** | **R** |
|---|---|---|---|---|
| C1 | 560 | 660 | 3,6 | 1 |
| C2 | 740 | 610 | 2,7 | 1 |
| C3 | 1020 | 590 | 1,96 | 1 |
| C4 | 860 | 610 | 2,32 | 0 |
| C5 | 775 | 660 | 2,8 | 0 |
| C6 | 615 | 680 | 3,25 | 1 |
| C7 | 665 | 540 | 3 | 1 |

| | | | | |
|---|---|---|---|---|
| C8 | 970 | 730 | 2,06 | 1 |
| C9 | 115 | 150 | 17,4 | 1 |
| C10 | 1115 | 1120 | 1,8 | 1 |

Valorile funcției de adaptare calculată folosind relația matematică (A3.6), a cromozomilor primei populații sunt:

$F_1 = 0{,}07$, $F_2 = 0{,}09$, $F_3 = 0{,}13$, $F_4 = 0{,}11$, $F_5 = 0{,}1$, $F_6 = 0{,}082$, $F_7 = 0{,}09$, $F_8 = 0{,}13$, $F_9 = 0{,}015$, $F_{10} = 0{,}15$.

Analiza rezultatelor obținute ne indică faptul cinci dintre cei 10 cromozomi au funcția de adaptare peste 0,1 și restricția 1, astfel acești cromozomi C2, C3, C7, C8 și C10 vor fi folosiți mai departe în următoarea populație. Trebuie specificat că doi (C4 și C5) dintre cei 10 cromozomi au fost eliminați de la început, din cauza faptului că valoarea lui R a fost 0.

Această metodă de selecție a cromozomilor care sunt folosiți la pasul următor se numește metoda elitistă.

### *A3.4. Generarea unei noi populații*

Noua populație (mulțime de soluții) va fi formată din cei cinci cei mai adaptați cromozomi din populația veche, plus alți cinci cromozomi noi, care vor fi obținuți din cei din populația anterioară la care se folosesc operatorii genetici de împerechere și mutație. Noua populație este ilustrată în fig. 3 cuprinde C2, C3, C7, C8 și C10, care vor deveni C1-C5, iar C6-C10 vor fi cromozomi noi obșinuți astfel:

- C6 și C7 – doi copii ai lui C2 și C3,
- C8 și C9 – doi copii ai lui C8 și C10,
- C10 – mutația a lui C7.

| | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | E10 | E11 | E12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| C2 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| C3 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| C4 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| C5 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| C6 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C7 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| C8 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| C9 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| C10 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |

**Fig. A3.3.** Noua populație de indivizi. C - cromozom

Copiii sunt obținuți prin operatorul genetic de împerechere în două puncte, mai exact genele 1-3 și 10-12 sunt păstrați de la un părinte, iar genele 4-9 de la celălalt părinte. Astfel, de la doi părinți se pot obține doi copii.

Copiii rezultați prin operatorul genetic de mutație se obțin prin schimbarea valorii unei gene din 1 în 0, sau din 0 în 1. Mutația realizată s-a efectuat asupra genei 3.

Noua populație este evaluată folosind aceeași metodă, adică prin determinarea valorilor funcțiilor obiectiv și a funcției de adaptare. Rezultatele obținute sunt prezentate în tabelul A3.3.

**Tabelul A3.3.** Valorile funcțiilor obiectiv pentru noua populație

| **Cromozom** | **Fo1** | **Fo2** | **Fo3** | **R** |
|---|---|---|---|---|
| C1 | 740 | 610 | 2,7 | 1 |
| C2 | 1020 | 590 | 1,96 | 1 |
| C3 | 665 | 540 | 3 | 1 |
| C4 | 970 | 730 | 2,06 | 1 |
| C5 | 1115 | 1120 | 1,8 | 1 |
| C6 | 610 | 490 | 3,27 | 1 |
| C7 | 705 | 420 | 2,83 | 1 |
| C8 | 1360 | 920 | 1,47 | 1 |
| C9 | 1130 | 930 | 1,76 | 1 |
| C10 | 865 | 740 | 2,31 | 1 |

Valorile funcției de adaptare calculată folosind relația matematică (A3.6), a cromozomilor primei populații sunt:

$F_1 = 0{,}08$, $F_2 = 0{,}11$, $F_3 = 0{,}07$, $F_4 = 0{,}1$, $F_5 = 0{,}12$, $F_6 = 0{,}065$, $F_7 = 0{,}75$, $F_8 = 0{,}148$, $F_9 = 0{,}123$, $F_{10} = 0{,}094$.

Valorile restricției sunt 1 pentru toți cromozomi.

Acest proces de formarea de noi populații este repetat până la un număr maxim de generații, de obicei între 100 și 10000. În acest exemplu, procesul de generare de noi cromozomi / soluții s-a oprit la această populație.

Următorul pas este selecția celor mai adaptași cromozomi. În acest exemplu, cromozomii cei mai adaptați sunt C2, C4, C5, C8 și C9. După ordonarea în ordine descrescătoare a acestor cromozomi rezultă în C8, C9, C5, C2 și C4. Pentru aceste cromozomi, se calculează consumul de energie, rezultatele sunt prezentate în tabelul A3.4.

**Tabelul A3.4.** Cromozomii elitiști ai ultimei generații

| **Nr. ordine** | **Cromozom** | **Adaptare** | **Consum MWh** | **Observație** |
|---|---|---|---|---|
| 1 | C8 | 0,148 | 32,45 | Punctajul cel mai mare |
| 2 | C9 | 0,123 | 39,28 | Mediu |
| 3 | C5 | 0,12 | 48,38 | Consumul cel mai mare |
| 4 | C2 | 0,11 | 27,75 | Consumul cel mai mic |
| 5 | C4 | 0,1 | 36,09 | Punctajul cel mai mic |

Din analiza tabelului A3.4 se observă că există trei posibile soluții în funcție de ceea ce se dorește. Astfel, dacă se pune accent pe contract, atunci varianta cea mai bună este C8, iar soluția este punerea în funcțiune a echipamentelor 3, 5, 6, 8, 9, și 12. În cazul în care se dorește consumul minim, atunci cromozomul C2 este soluția problemei, iar echipamentele 1, 2, 5, 6, 9 și 10 vor fi puse în funcțiune. Ultima variantă este C9, care are punctajul al doilea, și consumul al doilea ca mărime. În această situație, se vor pune în funcțiune echipamentele 2, 3, 6, 9, 11 și 12.