PROCESAREA CUNOȘTINȚELOR ȘI CALCUL INTELIGENT - PCCI

Algoritmul PSO.
Optimizări multiciriteriale

Cuprins:

- Algoritmul Particle Swarm Optimization (PSO)
- Optimizări multicriteriale. Fronturi Pareto
- Algoritmul Multi-Objective Swarm Optimizer (MOPSO)

Optimizarea deplasării roiurilor de particule (Particle Swarm Optimization, PSO)

Optimizarea deplasării roiurilor de particule

- Algoritmul PSO face parte din categoria algoritmelor de calcul evolutiv şi a fost conceput de John Kennedy şi Russell Eberhardt în 1995.
  - Lucrarea J. Kennedy, R. C. Eberhart, "Particle Swarm Optimization", Proceedings of IEEE International Conference on Neural Networks. Piscataway, NJ: IEEE Service Center, 1995: pp. 1942-1948

- Încearcă să simuleze matematic coregrafia graţioasă, dar imprevizibilă a stolurilor de păsări sau a bancurilor de peşti, care fac mişcări sincrone „ca la comandă"

- Pe parcursul dezvoltării algoritmului, autorii au constatat că modelul conceptual rezultat este unul de optimizare.

Optimizarea deplasării roiurilor de particule

- Asemenea AG, PSO iniţializează aleatoriu o populaţie de soluţii alcătuită din valori numerice sau şiruri de numere, numită swarm (roi), pe care o optimizează într-un proces iterativ.

- Spre deosebire de AG, fiecărei soluţii, numită particulă, îi este asociată o **viteză** inițială aleatorie, folosită pentru a o deplasa în spaţiul de căutare.

- Pentru fiecare particulă, algoritmul PSO reţine permanent cea mai bună poziţionare atinsă într-o valoare personal best (*pbest*, optim individual), pentru care se memorează particula propriu-zisă şi valoarea funcţiei sale de adaptare.

- De asemenea, este memorată şi cea mai bună poziționare a celei mai bune particule din roi, aşa-numitul global best (*gbest*, optim global sau lider).

Optimizarea deplasării roiurilor de particule

- Algoritmul PSO evoluează iterativ. Într-o iteraţie, fiecare particulă îşi schimbă viteza (accelerează) simultan către cea mai bună poziţie a sa (*pbest*) şi cea mai bună poziţie identificată de întregul roi (*gbest*). Acceleraţia este ponderată aleatoriu pe cele două direcţii.

- Există şi o versiune locală a PSO, în care, pe lângă *pbest*, fiecare particulă memorează şi o soluţie local best (*lbest*, optim local), obţinută de particulele aflate în vecinătatea particulei respective.

Algoritmul PSO de principiu

- Iniţializează o populaţie aleatoare de particule cu d variabile (dimensiuni), cărora li se asociază câte un vector aleatoriu de viteză, de dimensiune d;
- **repetă**
  - Pentru fiecare particulă,
    - Se calculează funcţia de adaptare
    - Se compară funcţia de adaptare curentă a particulei cu valoarea corespunzătoare optimului individual *pbest*. Dacă a fost găsită o valoare mai bună decât *pbest*, particula pbest anterioară este înlocuită de particula curentă, fiind memorată împreună cu funcţia ei de adaptare
    - Se compară funcţia de adaptare curentă a particulei cu valoarea optimă globală descoperită până în prezent de roi, *gbest*. Dacă a fost găsită o valoare mai bună, decât *gbest*, particula *gbest* (*lider*) memorată anterior este înlocuită de particula curentă, împreună cu funcţia ei de adaptare
    - Se actualizează viteza şi poziţia particulei, cu relaţiile:

$$v = v + c_1 \cdot random \cdot (pbest - particula) + c_2 \cdot random \cdot (gbest - particula)$$

$$particula = particula + v$$

- **până** se atinge un anumit număr de generaţii ori o valoare acceptabilă a funcţiei de adaptare a particulei *gbest*.

PSO - detalii de implementare

- Viteza fiecărui element al unei particule este limitată la o valoare maximă V_{max} specificată de utilizator. Dacă suma acceleraţiilor face particula să depăşească această viteză-limită, atunci viteza reală va fi limitată la V_{max}.

- V_{max} este un parametru important, deoarece determină rapiditatea deplasării particulelor, adică a explorării spaţiului de căutare. Dacă V_{max} este prea mare, particulele pot "zbura" pe lângă soluţii bune, în vreme ce dacă V_{max} este prea mică, particulele riscă să nu exploreze suficient de departe, dincolo de regiunile cu minime locale, rămânând blocate în zone de minim local.

PSO - detalii de implementare

- Constantele de accelerare c_1 şi c_2 reprezintă ponderile cu care o particulă este trasă către optimul ei individual şi către optimul global. Optimizarea lor reglează "tensiunea" din interiorul roiului.

- Valori mici ale acestor parametri permit particulelor să se îndepărteze mult de zona-ţintă înainte de a fi aduse înapoi, în vreme ce valori mari provoacă mişcări rapide înspre zonele-ţintă.

- Încercările efectuate de-a lungul timpului au descoperit că valorile optime pentru c_1 şi c_2 sunt de cele mai multe ori egale, având valoarea 2.
- V_{max} rămâne singurul parametru reglabil, în intervalul maxim de 10-20% din domeniul de căutare al fiecărei variabile din componenţa particulei.

PSO - detalii de implementare

- Dimensiunea optimă a populaţiei iniţiale depinde de problema rezolvată.
- O valoare orientativă recomandată este de 20-50 de indivizi.
- În general, algoritmul PSO funcţionează optim pe populaţii mai mici decât alte algoritme evolutive, precum AG.

PSO - detalii de implementare

- Viteza V_{max} care controlează rapiditatea de mişcare a roiului este înlocuită în implementările actuale ale PSO cu un termen de inerţie, care oferă performanţe îmbunătăţite prin corectarea vitezei v din ecuaţiile standard de actualizare ale particulelor, care se rescriu:

$$v = w \cdot v + c_1 \cdot random \cdot (pbest - particula) + c_2 \cdot random \cdot (gbest - particula)$$
$$particula = particula + v$$

- În cea mai simplă implementare, inerţia w porneşte cu o valoare mare, care scade linear de-a lungul procesului iterativ.

- Valorile inițială şi finală recomandate sunt de 0,9, respectiv 0,4.

- Alegerea corectă a factorului de inerţie oferă un echilibru între explorarea globală şi cea locală.

PSO - algoritmul local

- În versiunea "locală" a PSO, particulele nu cunosc informaţii globale, ci numai informaţii despre ele şi despre particulele aflate în vecinătatea lor.

- În loc să se deplaseze către optimul personal *pbest* şi cel global *gbest*, ele se vor orienta către optimul individual *pbest* şi cel local *lbest*, al particulelor din vecinătate.

- De exemplu, dacă dimensiunea vecinătăţii este setată la 2, atunci particula *(i)* îşi va compara funcţia de adaptare cu particulele vecine *(i-1)* şi *(i+1)*, alese în funcţie de distanţa minimă.

PSO - algoritmul local

- La nivelul algoritmului general, implementarea variantei PSO locale necesită o singură schimbare în ecuaţiile de principiu ale actualizării particulelor, unde particula *gbest* este înlocuită cu particula *lbest*.

- Experimentele au arătat că o dimensiune optimă a vecinătăţii este de 15% din dimensiunea populației, adică, pentru o dimensiune a populaţiei de 40, o vecinătate de şase particule, trei de o parte şi trei de alta, este suficientă.

Aplicații ale PSO

- probleme de optimizare:
  - proiectarea sistemelor
  - optimizarea multicriterială
  - clasificarea, recunoaşterea formelor
  - modelarea sistemelor biologice
  - planificarea operaţiunilor
  - procesarea semnalelor
  - jocurile, aplicaţiile de robotică
  - luarea deciziilor
  - simularea.

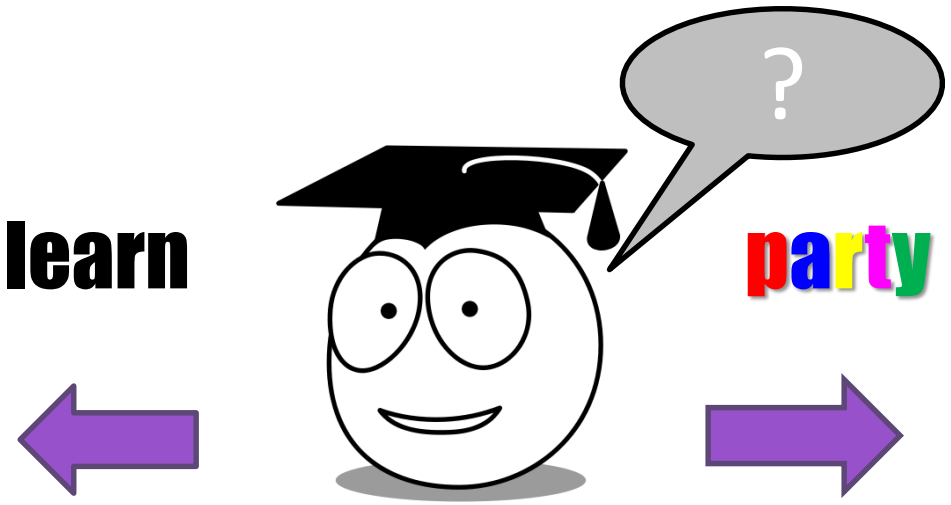
Aplicații ale PSO

- Exemple de aplicaţii realizate:
  - proiectarea controllerelor fuzzy
  - planificarea în timp real a traseului de deplasare a roboţilor
  - simularea semnalelor EEG
  - diagnoza arsurilor
  - recunoaşterea gesturilor
  - Identificarea ţintelor.

- Cele mai multe probleme din practică nu necesită optimizarea unui singur obiectiv, ci realizarea unui compromis între mai multe obiective, adesea conflictuale.

- Varianta multicriterială a algoritmului PSO este algoritmul MOPSO – Multi-Objective PSO.

- Mecanismul MOPSO este asemănător celui PSO, cu unele adaptări impuse de optimizarea multicriterială.



Optimizarea multicriterială. Fronturi Pareto

Optimizări multicriteriale. Fronturi Pareto

- În cazul optimizărilor multicriteriale, soluţia nu mai este unică.
- Ea rezultă ca un compromis între două sau mai multe obiective.

Atunci când optimizarea consideră două obiective conflictuale (când unul scade, celălalt creşte şi viceversa), se formează aş-anumitele fronturi Pareto de soluţii optime în care un punct reprezintă o valoare de compromis între cele două obiective.

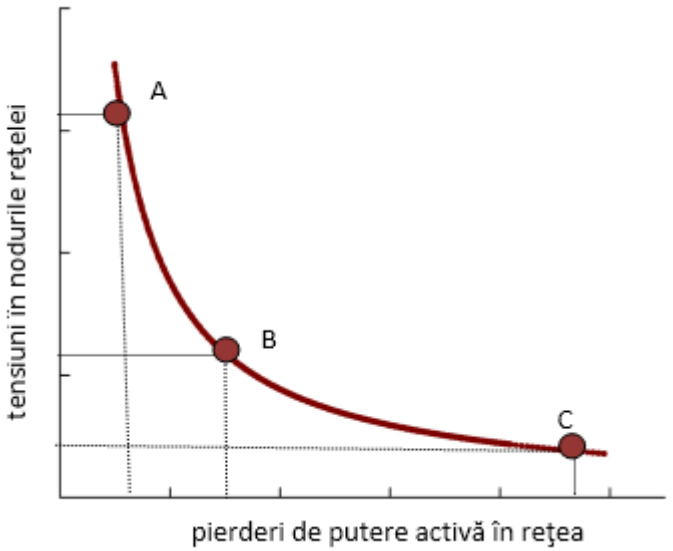
Exemplu de front Pareto

- Problemă teoretică: minimizarea pierderilor de putere activă pe laturi şi a tensiunii din nodurile unei reţele electrice. Pot exista trei tipuri de soluţii:

- (A) – pierderile de putere mici sunt un indicator al faptului că sarcina este mică în reţea. Când sarcina este mică, tensiunile din noduri au valori mari;

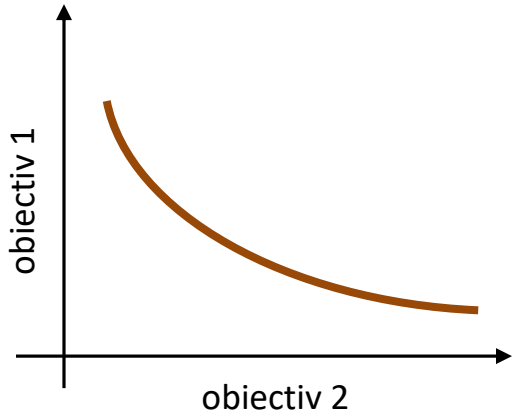
- (B) – pierderile de putere şi tensiunile au valori medii. Acestea sunt soluţii de compromis, atunci când niciuna dintre funcţiile obiectiv nu are valori extreme;

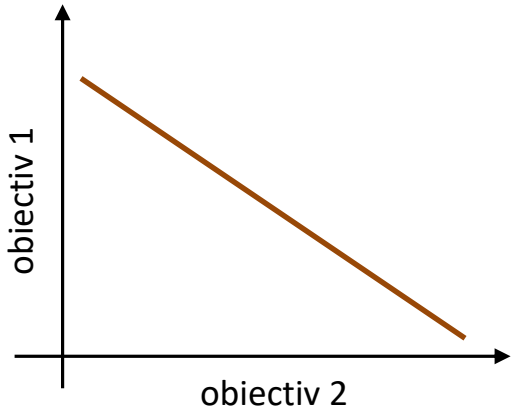
- (C) – pierderi mari de putere activă se înregistrează când încărcarea liniilor este mare (sarcina este mare în reţea). Când sarcina este mare, de obicei tensiunile nodale vor fi mai mici, deoarece căderile de tensiune pe laturi vor fi semnificative.



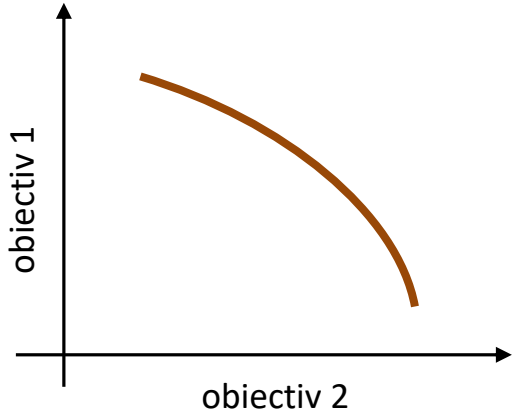
Forme tipice de fronturi Pareto

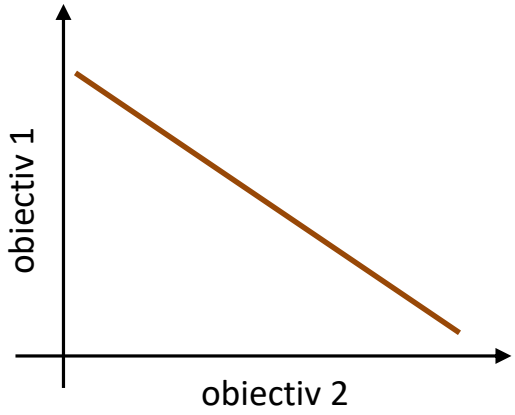
- Pentru probleme de minimizare





- Pentru probleme de maximizare





Fronturi Pareto

- În determinarea unui front Pareto, trebuie urmărite trei obiective principale:

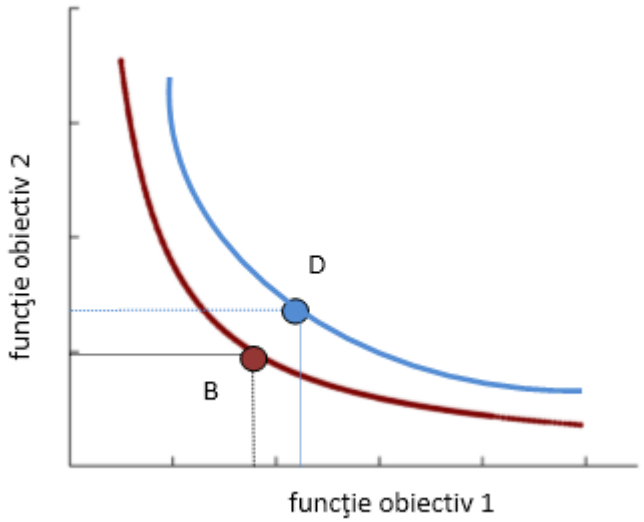
  - Maximizarea numărului de particule care intră în componenţa sa, pentru a avea la dispoziţie cât mai multe soluţii optime;

  - Frontul identificat de algoritm trebuie să fie cât mai apropiat de frontul Pareto optim pentru problema analizată.

  - Împrăştierea cât mai uniformă a particulelor pe lungimea frontului, pentru a avea disponibile cât mai multe soluţii pentru întreg intervalul de variaţie al celor două funcţii obiectiv.

Frontul Pareto optim

- Pentru o problemă dată, pot fi identificate mai multe fronturi Pareto, dintre care doar unul este cel optim;
- Pe frontul Pareto optim pot intra doar particule nedominate. Particulele de pe celelalte fronturi sunt dominate.
- O particulă este dominată dacă are simultan ambele funcţii obiectiv cu valori mai mari decât o altă particulă deja descoperită. În cazul prezentat în figură, particula D este dominată de particula B.



$$FO_{1,B} < FO_{1,D}$$

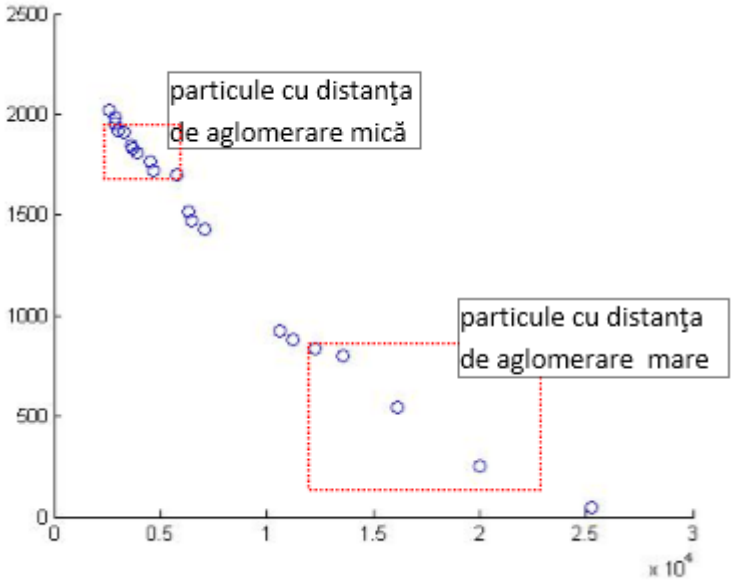
$$FO_{2,B} < FO_{2,D}$$

Uniformizarea fronturilor Pareto

- De obicei, pentru frontul Pareto optim se specifică un număr maxim de elemente.

- Dacă acest număr este depăşit, frontul este triat, eliminându-se particule din zona cea mai aglomerată a frontului, pentru a da posibilitatea populării zonelor mai rarefiate.

- Sortarea şi eliminarea particulelor se face după calculul unei distanţe de aglomerare.



Optimizarea deplasării roiurilor de particule pentru probleme multiobiectiv (Multi-Objective Particle Swarm Optimization, MOPSO)

Algoritmul MOPSO vs algoritmul PSO

- Într-o optimizare cu un singur obiectiv (cu algoritmul PSO), soluţia este unică.

- Rezultatul optimizării multiobiectiv (cu algoritmul MOPSO) este un front Pareto cu un număr de soluţii considerate la fel de bune, deoarece sunt nedominate.

- Aceste soluţii sunt descoperite treptat, pe parcursul procesului evolutiv, şi sunt păstrate într-o populaţie paralelă, numită arhivă externă (external archive).

- Dacă pe parcursul evoluţiei se descoperă cu populaţia de lucru soluţii noi care domină soluţii deja existente în arhiva externă, atunci cele din urmă, dominate, sunt eliminate din arhiva externă, care va conţine în final frontul Pareto optim căutat.

Algoritmul MOPSO vs algoritmul PSO

- În algoritmul PSO standard, cu o singură funcţie obiectiv, particula lider (*global best* sau *gbest*) este unică în fiecare iteraţie, fiind cea mai bună particulă găsită până în momentul respectiv al procesului de optimizare.

- În cazul algoritmului MOPSO, selecţia particulei lider se reia într-o iteraţie pentru fiecare particulă din roi, iar noul lider este ales din arhiva externă.

Alegerea liderului în algoritmul MOPSO

- Cele mai simple variante de alegere a liderului sunt alegerea aleatorie sau folosirea regulii ruletei de la algoritmele genetice.

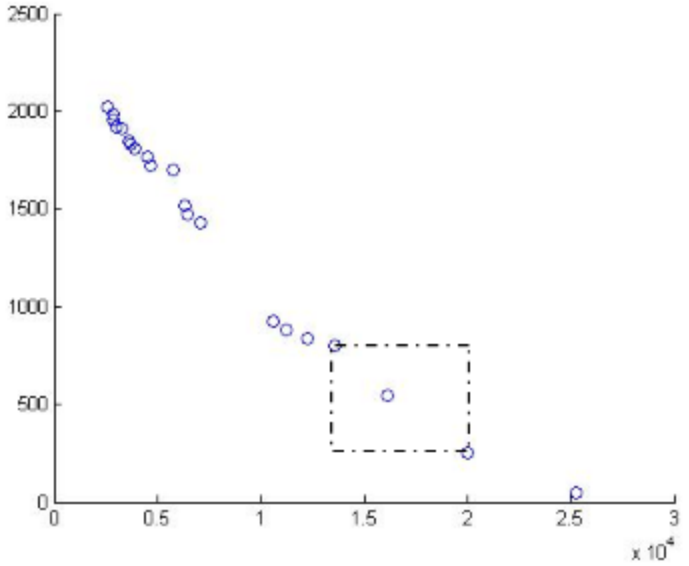
- Alte variante utilizează criteriul densității, favorizând alegerea liderilor din zonele cele mai rarefiate ale frontului Pareto, pentru a favoriza popularea acelor zone cu particule noi:
  - Criteriul distanţei de aglomerare (crowding distance criterion)
  - Criteriul indexului de aglomerare sau al nişei (niche creiterion)

- Se pot realiza şi combinaţii între aceste metode.

Alegerea liderului după distanța de aglomerare

- Metoda distanţei de aglomerare măsoară distanţa geometrică dintre fiecare particulă şi cei mai apropiaţi doi vecini ai săi.

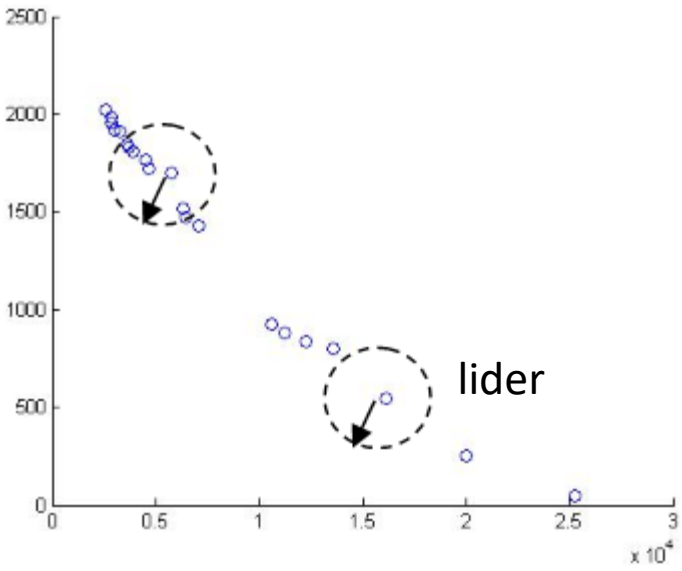
- Este aleasă lider particula cu distanţa de aglomerare cea mai mare (din zona cea mai rarefiată a frontului), pentru a încuraja popularea uniformă a frontului Pareto



Alegerea liderului după indexul de aglomerare

- Pentru fecare particulă, se numără particulele vecine aflate în interiorul unei suprafeţe circulare de o anumită rază numită nişă

- Este aleasă lider particula cu indexul de aglomerare cel mai mic, cu cei mai puţini vecini, adică din zona cea mai rarefiată a frontului, pentru a încuraja popularea uniformă a frontului Pareto



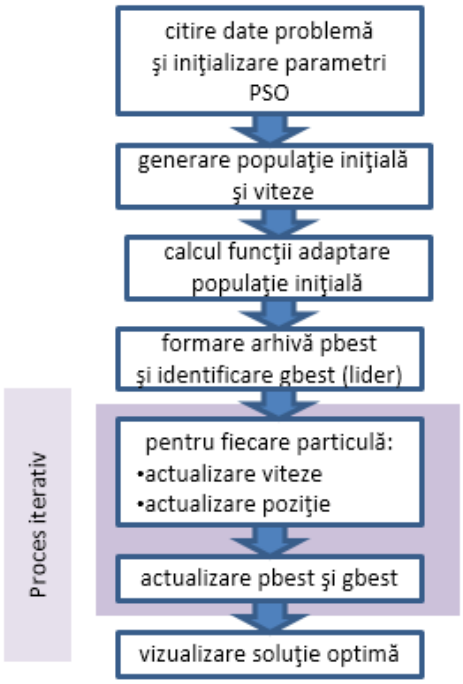
Algoritmul MOPSO vs algoritmul PSO

- Mecanismul de actualizare al populaţiei este acelaşi cu al algoritmului PSO, cu particularitatea că particula gbest este înlocuită de lider.

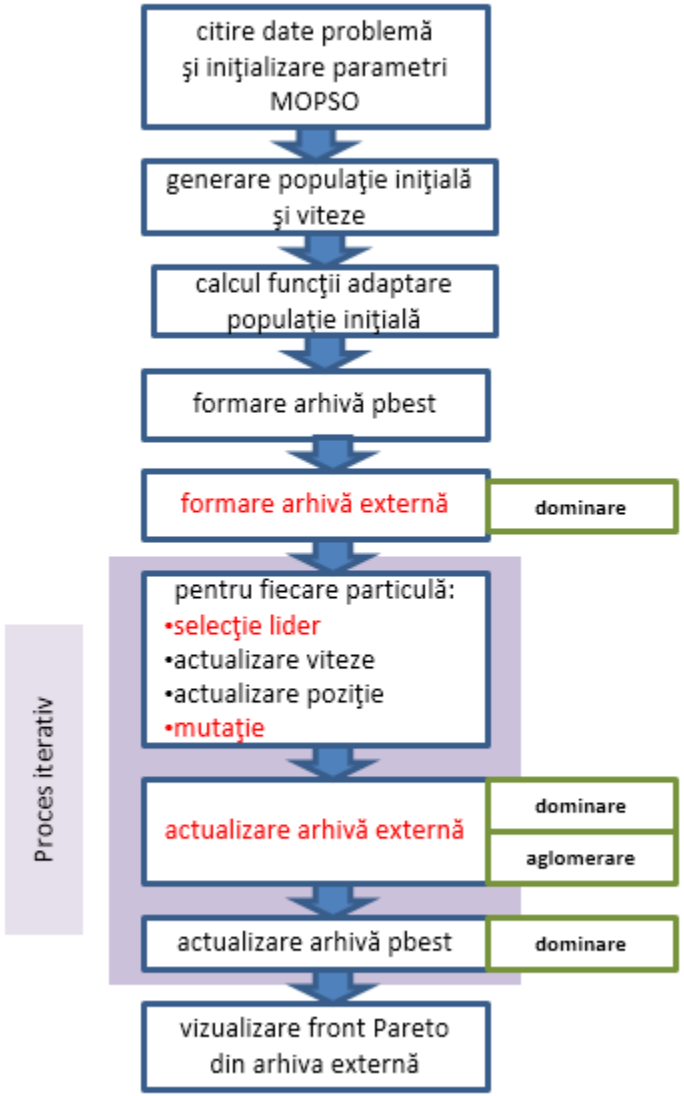
$$v = w \cdot v + c_1 \cdot random \cdot (pbest - particula) + c_2 \cdot random \cdot (lider - particula)$$
$$particula = particula + v$$

- Un fenomen des întâlnit este stagnarea populaţiei după un număr de iteraţii, când mecanismul de actualizare al vitezei nu mai reuşeşte să descopere soluţii mai bune. Pentru depăşirea acestei dificultăţi, algoritmul MOPSO standard este prevăzut cu un operator de mutaţie asemănător celui utilizat de algoritmele genetice, care trebuie adaptat în funcţie de problema de optimizare rezolvată.

Algoritmul MOPSO vs algoritmul PSO



PSO



MOPSO

Particularități ale algoritmului PSO

- Când se folosesc populaţii mari, există pericolul ca arhiva externă să se aglomereze foarte repede.

- Pentru evitarea acestei situaţii, de obicei se impune un număr maxim admis de particule pentru arhiva externă.

- Când numărul de soluţii nedominate descoperite depăşeşte această limită, se aplică un criteriu de departajare precum calculul distanţei sau indexului de aglomerare şi se elimină particule din zonele cele mai dense ale frontului Pareto.

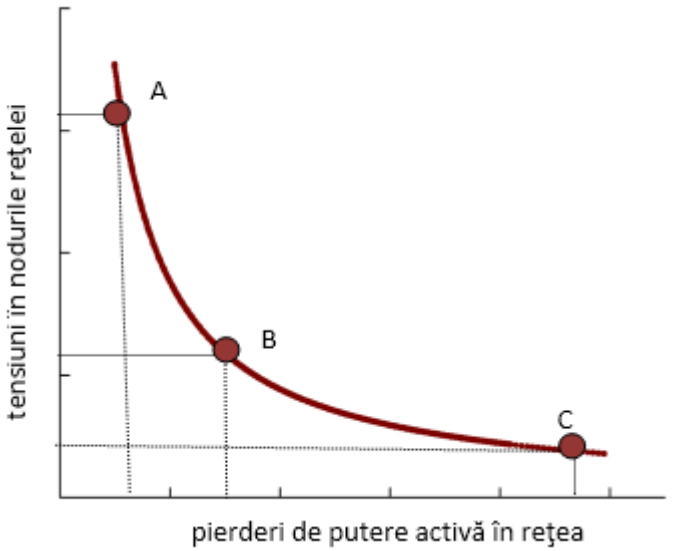
Exemplu de aplicare al algoritmului MOPSO

- Problemă teoretică: minimizarea pierderilor de putere activă pe laturi şi a tensiunii din nodurile unei reţele electrice. Pot exista trei tipuri de soluţii:

- (A) – pierderile de putere mici sunt un indicator al faptului că sarcina este mică în reţea. Când sarcina este mică, tensiunile din noduri au valori mari;

- (B) – pierderile de putere şi tensiunile au valori medii. Acestea sunt soluţii de compromis, atunci când niciuna dintre funcţiile obiectiv nu are valori extreme;

- (C) – pierderi mari de putere activă se înregistrează când încărcarea liniilor este mare (sarcina este mare în reţea). Când sarcina este mare, de obicei tensiunile nodale vor fi mai mici, deoarece căderile de tensiune pe laturi vor fi semnificative.



Exemplu de aplicare al algoritmului MOPSO

- studiu de optimizare a configurației de funcţionare a reţelelor radiale de distribuţie a energiei electrice din punctul de vedere al două obiective conflictuale, alese dintre trei disponibile:
  - Minimizarea pierderilor de putere şi energie activă în funcţionare (dP)
  - Reducerea puterii nelivrate consumatorilor pe o anumită perioadă de timp, în cazul producerii defectelor (PNS – power not supplied)
  - Reducerea nivelului tensiunii în reţea în regimurile de sarcină redusă (dU)

- restricţii:
  - configuraţiile valide trebuie să fie obligatoriu radiale
  - toţi consumatorii trebuie să fie alimentaţi

- rezultatul: un front de soluţii oprime Pareto

Funcțiile obiectiv

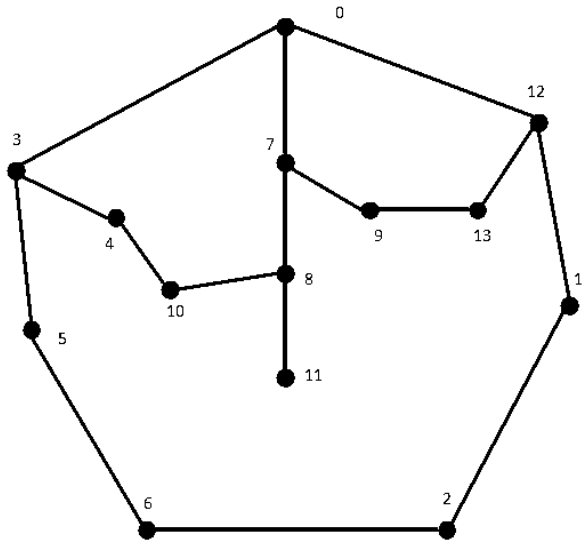
$$dP = \sum_{h=1}^{24} \sum_{i=1}^{nr\ laturi} dP_i$$

$$PNS = \sum_{h=1}^{24} \left(\sum_{j=1}^{nr\ laturi} \sum_{k=1}^{nod\ nealim} \lambda_j \cdot P_{jk} \right) = \sum_{h=1}^{24} \left(\sum_{j=1}^{nr\ laturi} \lambda_j \cdot \sum_{k=1}^{nod\ nealim} P_{jk} \right)$$

$$dU = \sum_{h=1}^{24} \sum_{i=1}^{nr\ noduri} U_i$$

R_i – rezistenţa laturii i din reţea
I_i – curentul care circulă prin latura i
λ_j - intensitatea de defectare a laturii j (numărul de defectări într-o perioadă de zece ani)
P_{jk} – puterea nealimentată în toate nodurile k de după latura j
dP_i - pierderile de putere pe latura i
U_i - modulul tensiunii din nodul i
h – oră din zi

Rețeaua electrică pentru studiul de caz

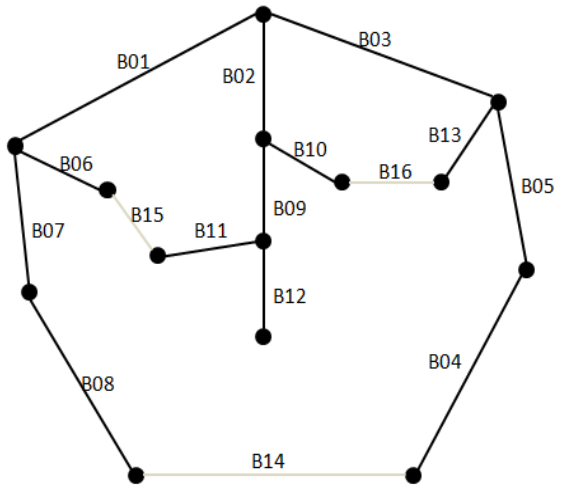


Graful asociat schemei monofilarre

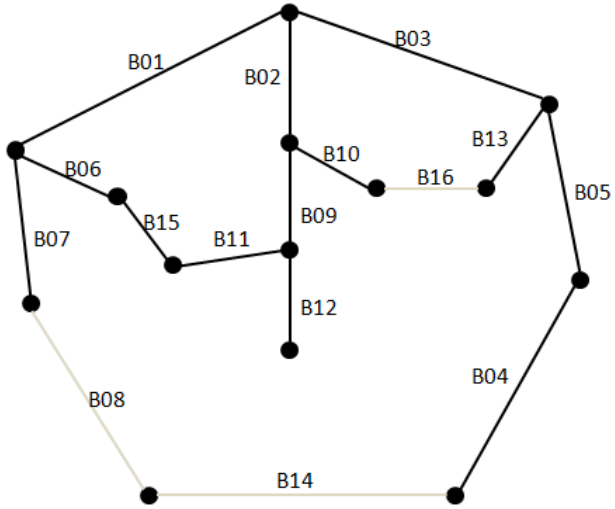
- 20 kV tensiune nominală
- 14 noduri
- 16 laturi
- Pentru obţinerea configuraţiilor radiale, trebuie debuclate 3 laturi
- Sarcinile din noduri repretentate prin curbe de sarcină pe 24 de ore

Codificarea soluțiilor

- Vectori cu numere întregi, reprezentând laturile deconectate, combinaţii care trebuie validate.



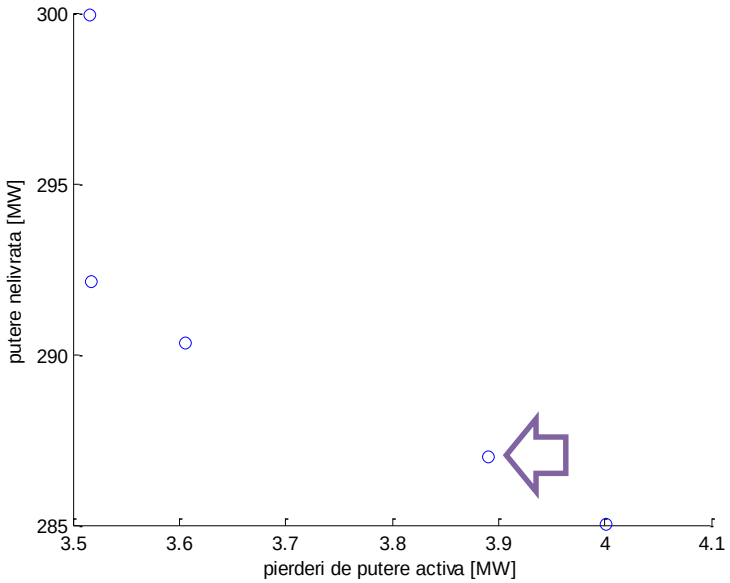
Particula [14,15,16] – validă
(reţea radială, toţi consumatorii sunt alimentaţi)

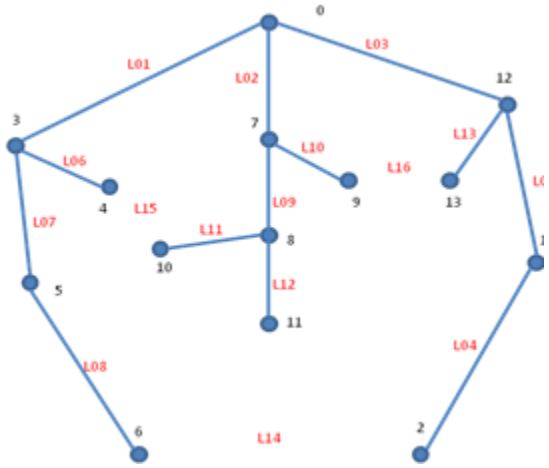


Particula [8,14,16] – invalidă
– există o buclă şi un consumator nealimentat

Rezultate pentru funcțiile obiectiv dP - PNS

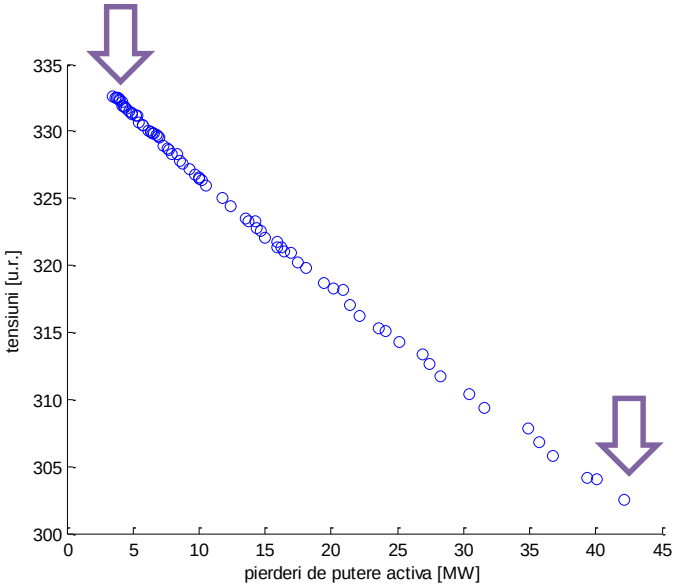
Frontul Pareto optim conţine
5 soluţii

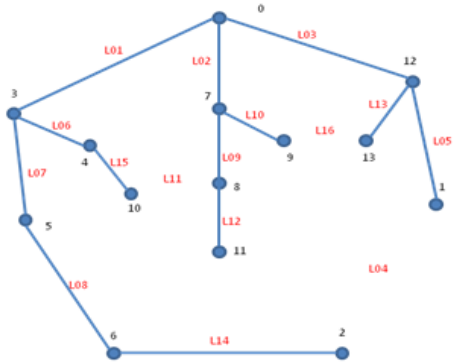




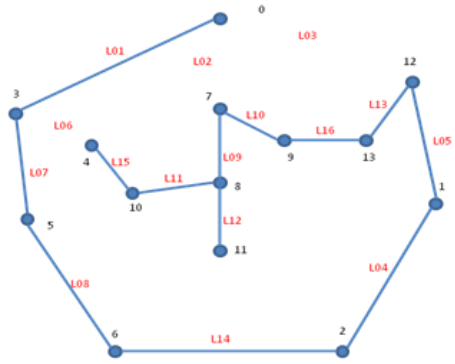
dP=3,89 MW
PNS=286,99 MW
Laturi deconectate: 16, 14, 15

Rezultate pentru funcțiile obiectiv dP - dU



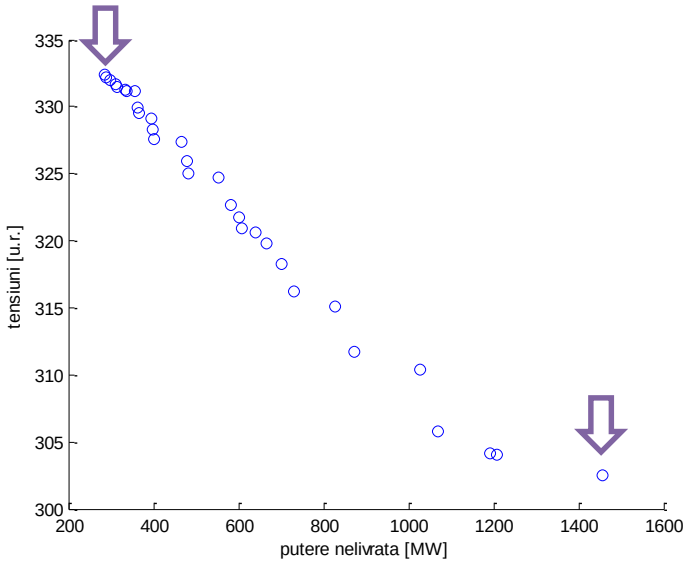


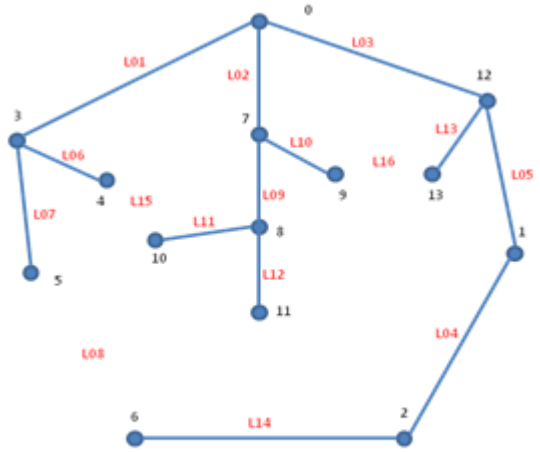
dP=3,517 MW; dU=332,65 MW;
Laturi deconectate: 16, 4, 11



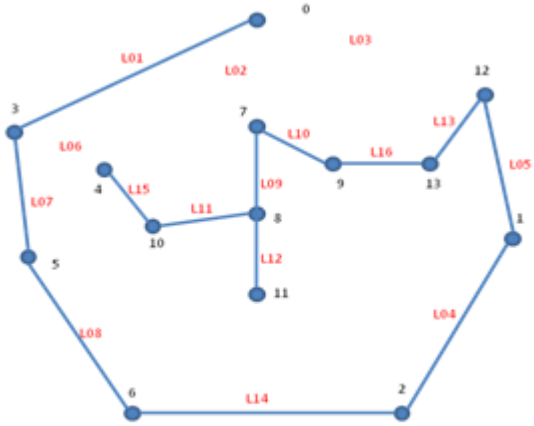
dP=42,18 MW; dU=302,47 MW;
Laturi deconectate: 2, 6, 3

Rezultate pentru funcțiile obiectiv PNS - dU





PNS=285,02 MW; dU=332,41 MW;
Laturi deconectate: 16, 8, 15

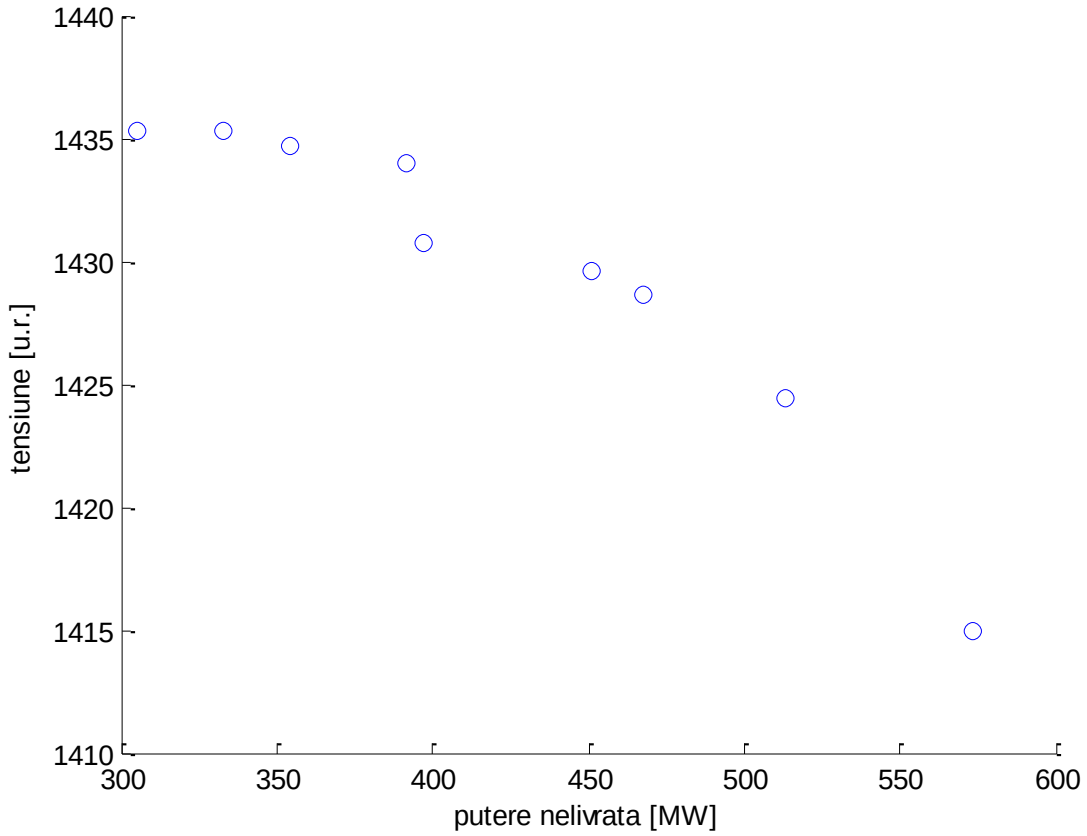


PNS=1457,13 MW; dU=302,47 MW;
Laturi deconectate: 2, 6, 3

Exemplu 2:

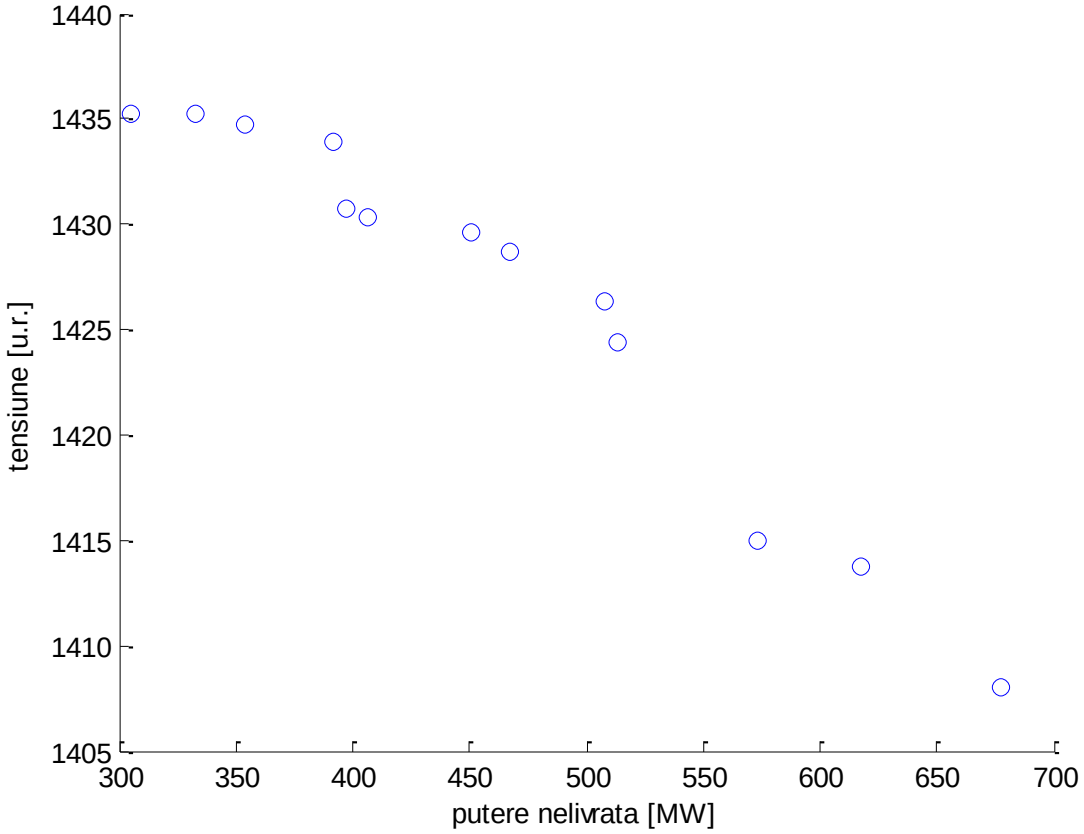
- Evoluţia unui front Pareto în timpul unui proces iterativ

Evoluția unui front Pareto



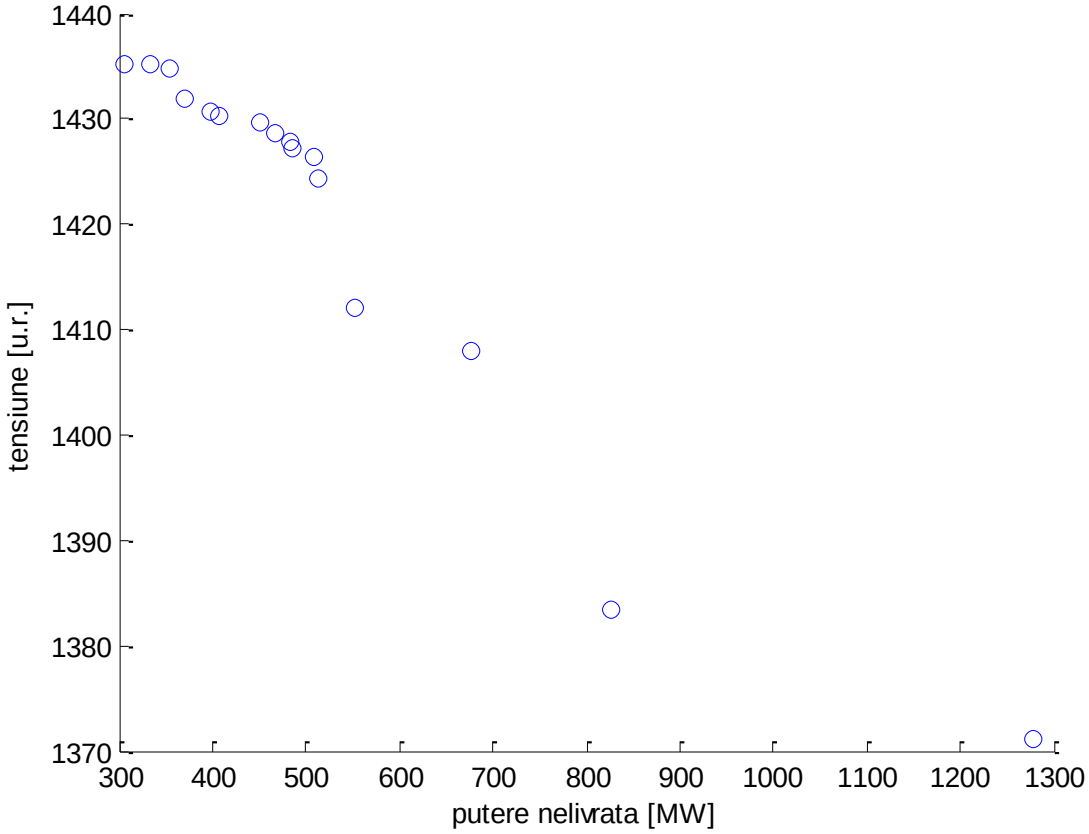
iteraţia 1

Evoluția unui front Pareto



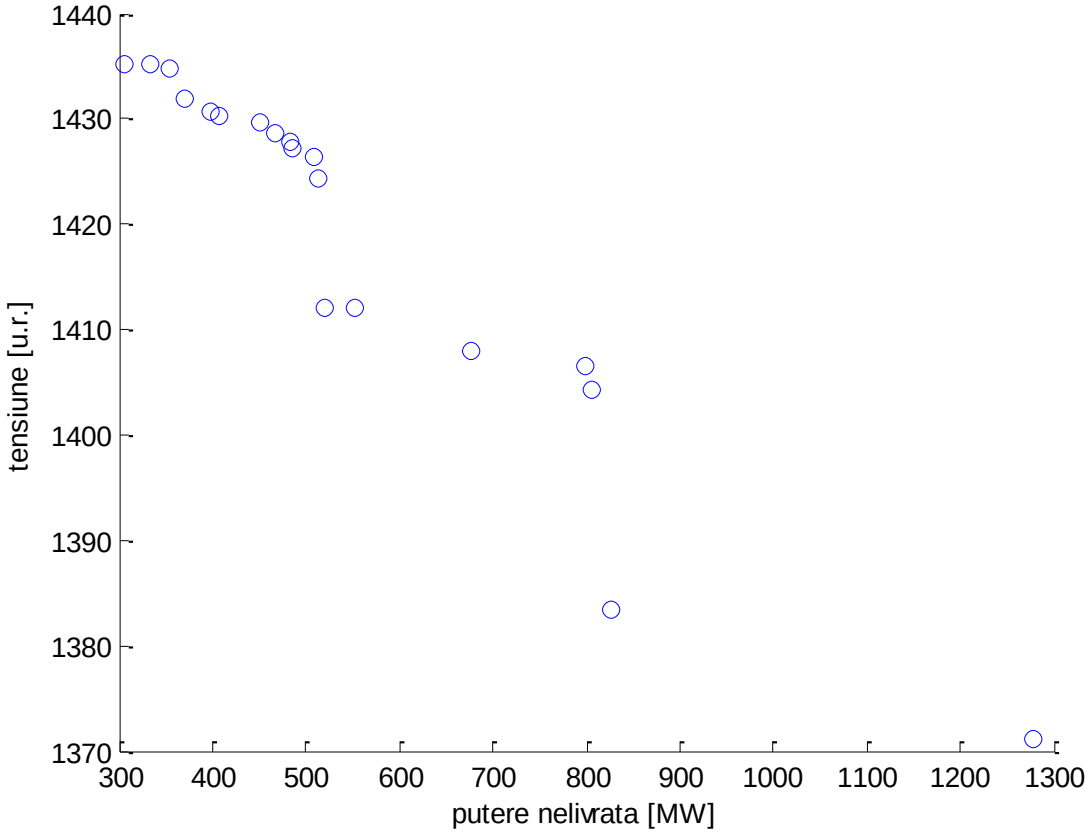
iteraţia 2

Evoluția unui front Pareto



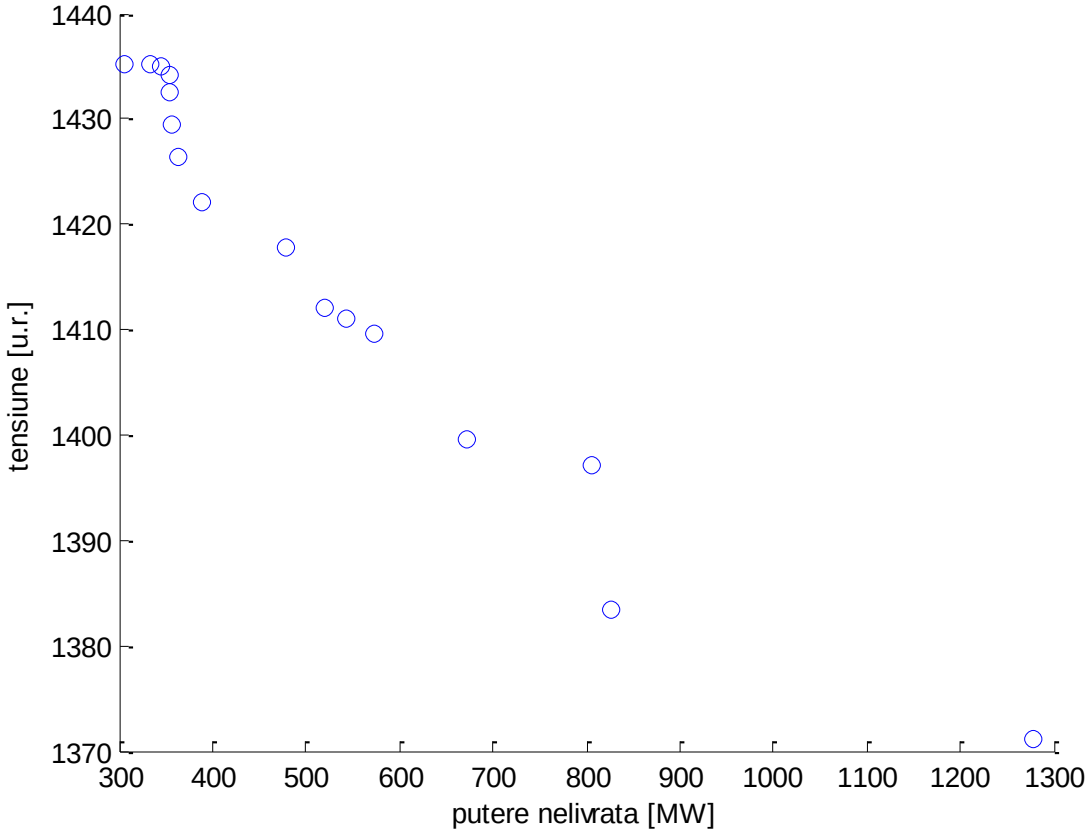
iteraţia 3

Evoluția unui front Pareto



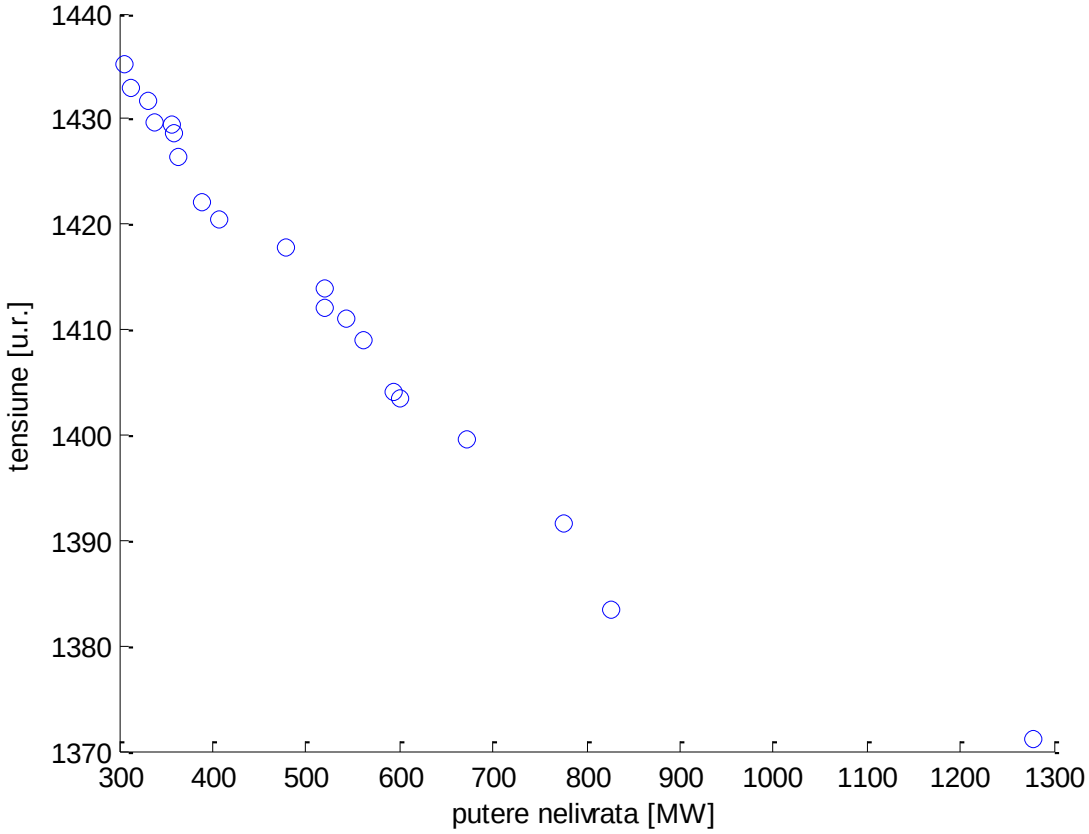
iteraţia 4

Evoluția unui front Pareto



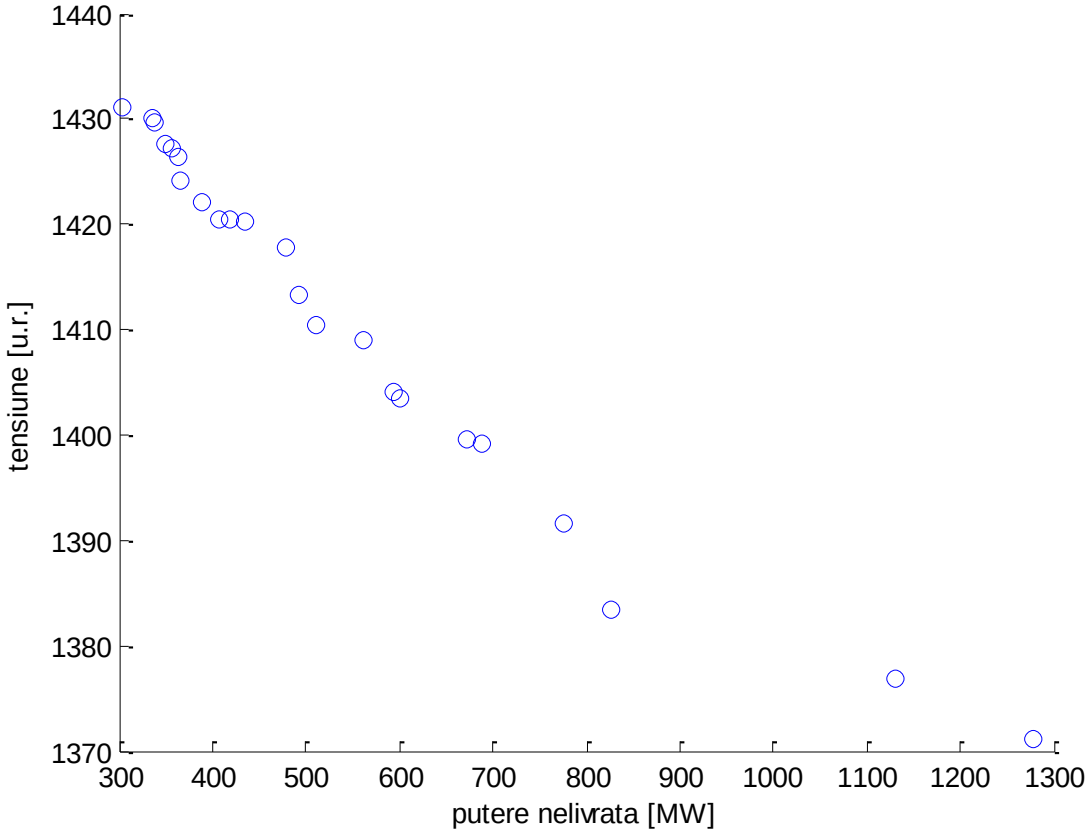
iteraţia 20

Evoluția unui front Pareto



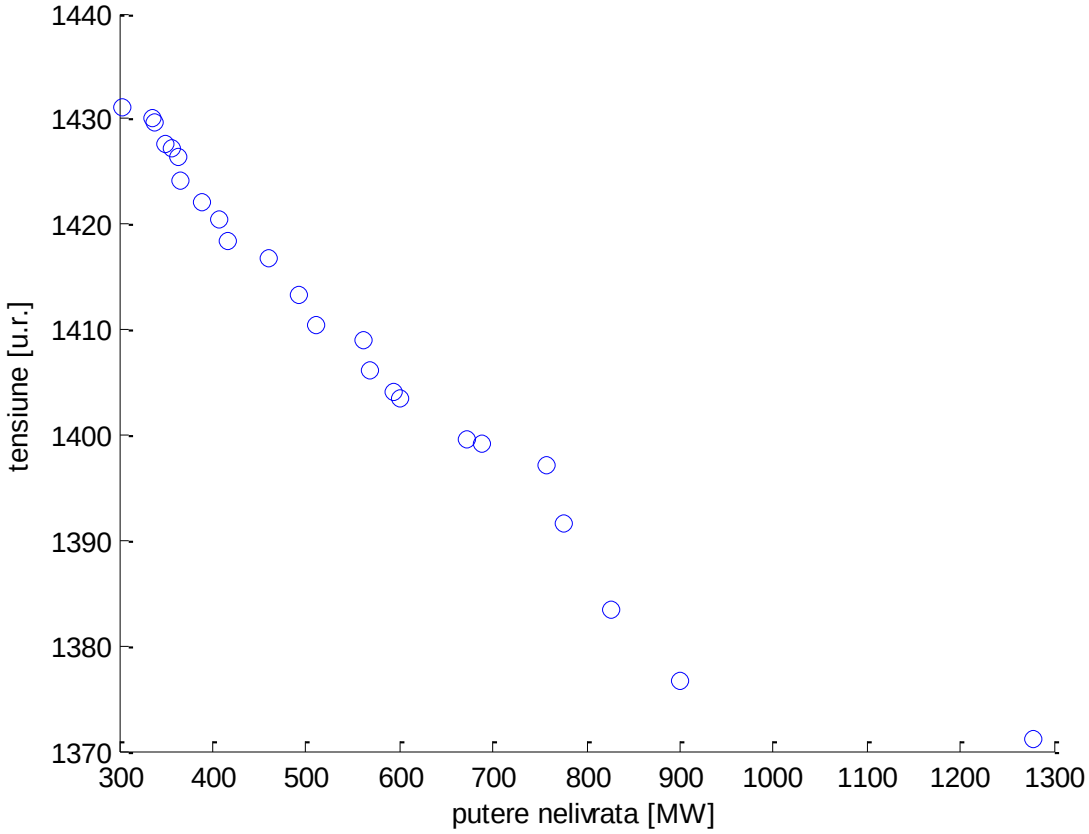
iteraţia 40

Evoluția unui front Pareto



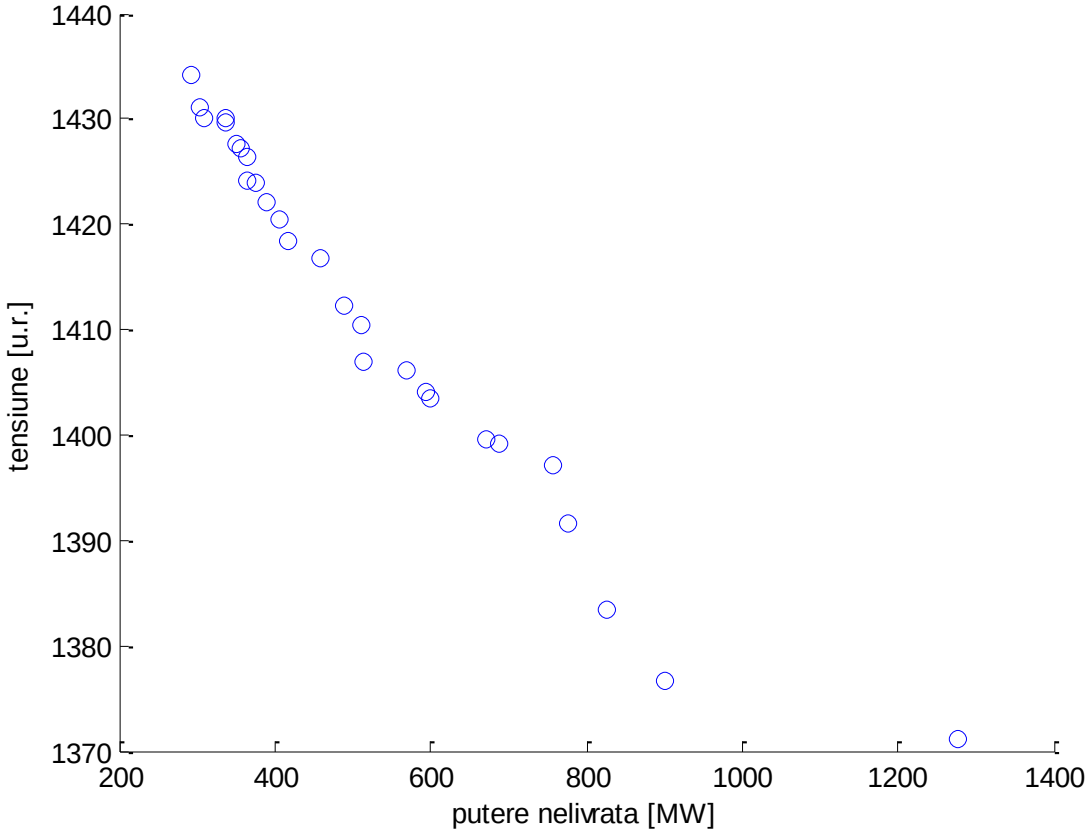
iteraţia 60

Evoluția unui front Pareto



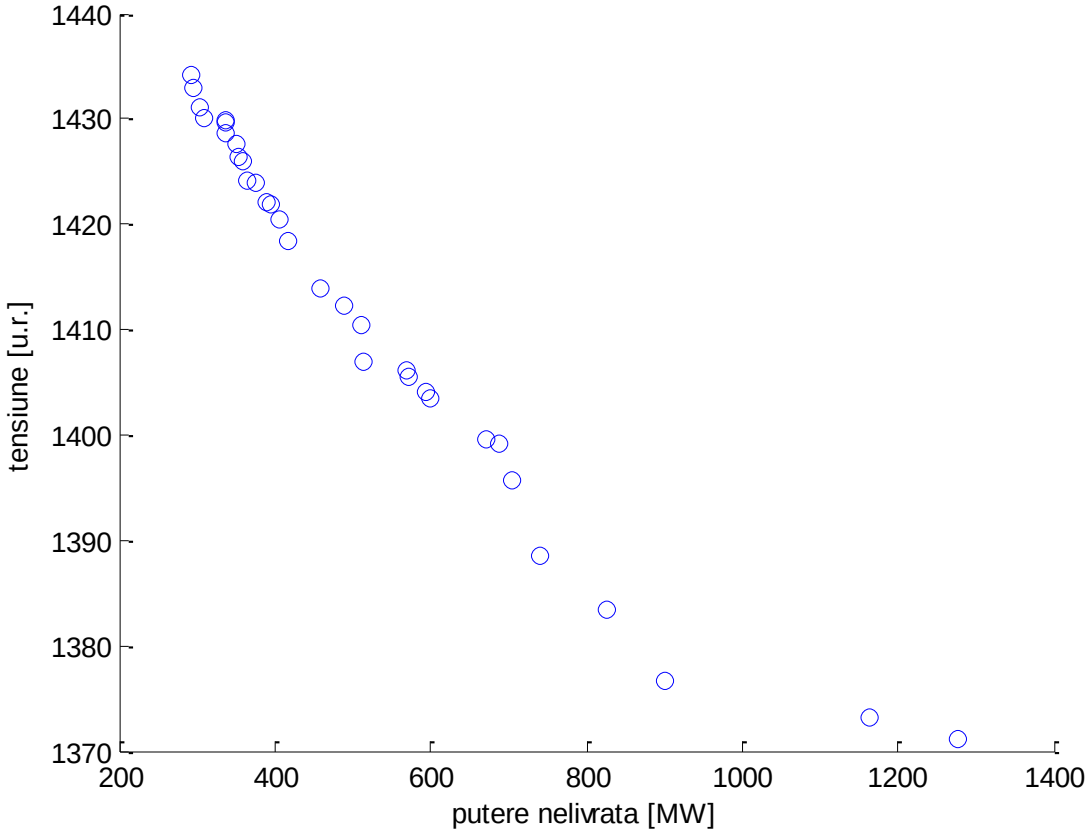
iteraţia 80

Evoluția unui front Pareto



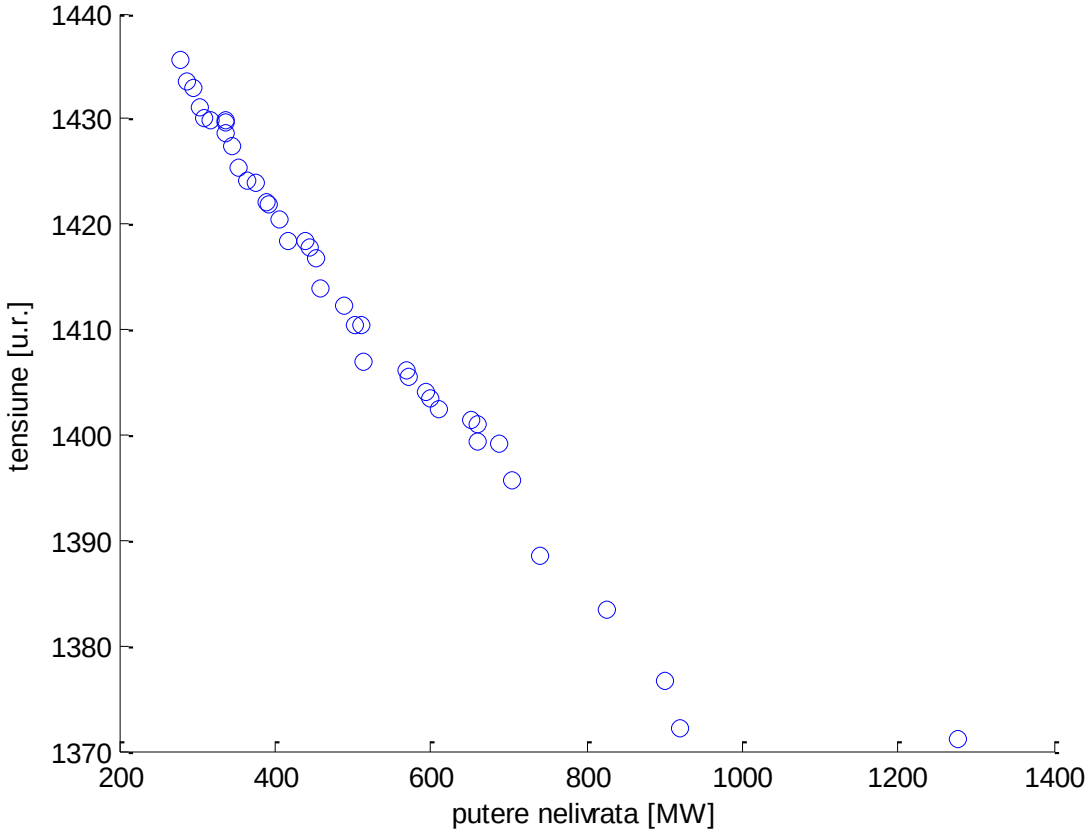
iteraţia 100

Evoluția unui front Pareto



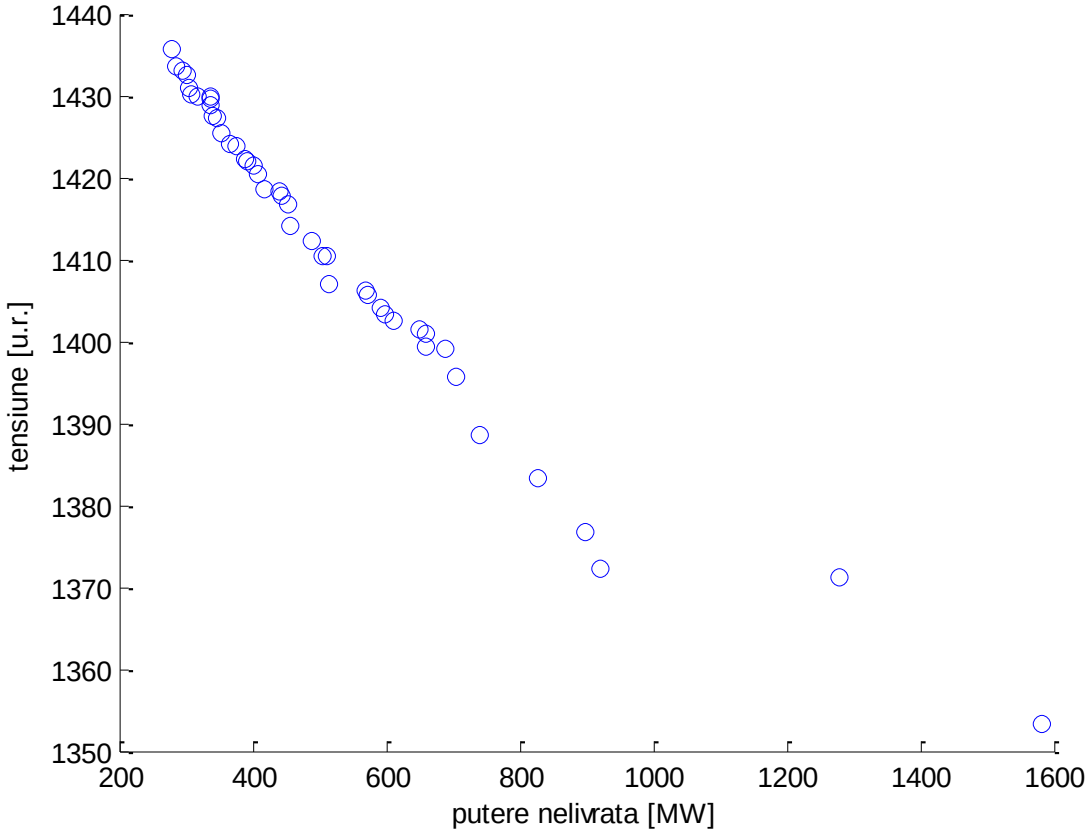
iteraţia 200

Evoluția unui front Pareto



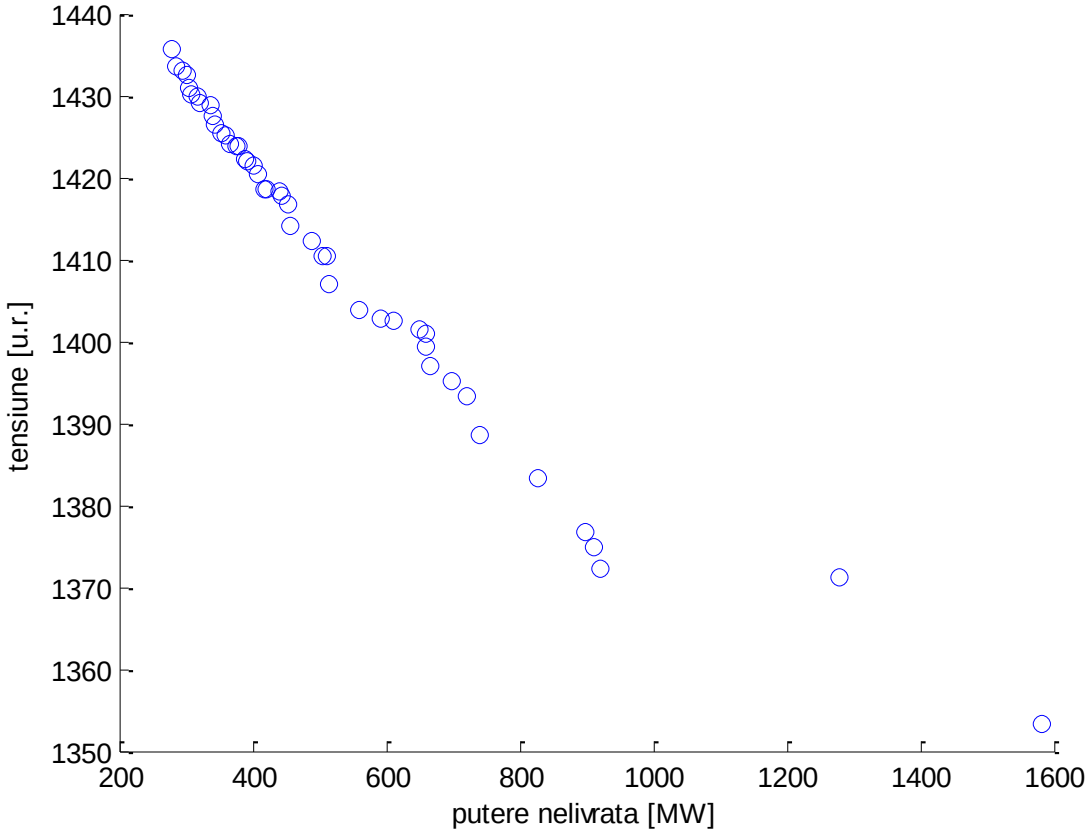
Iteraţia 300

Evoluția unui front Pareto



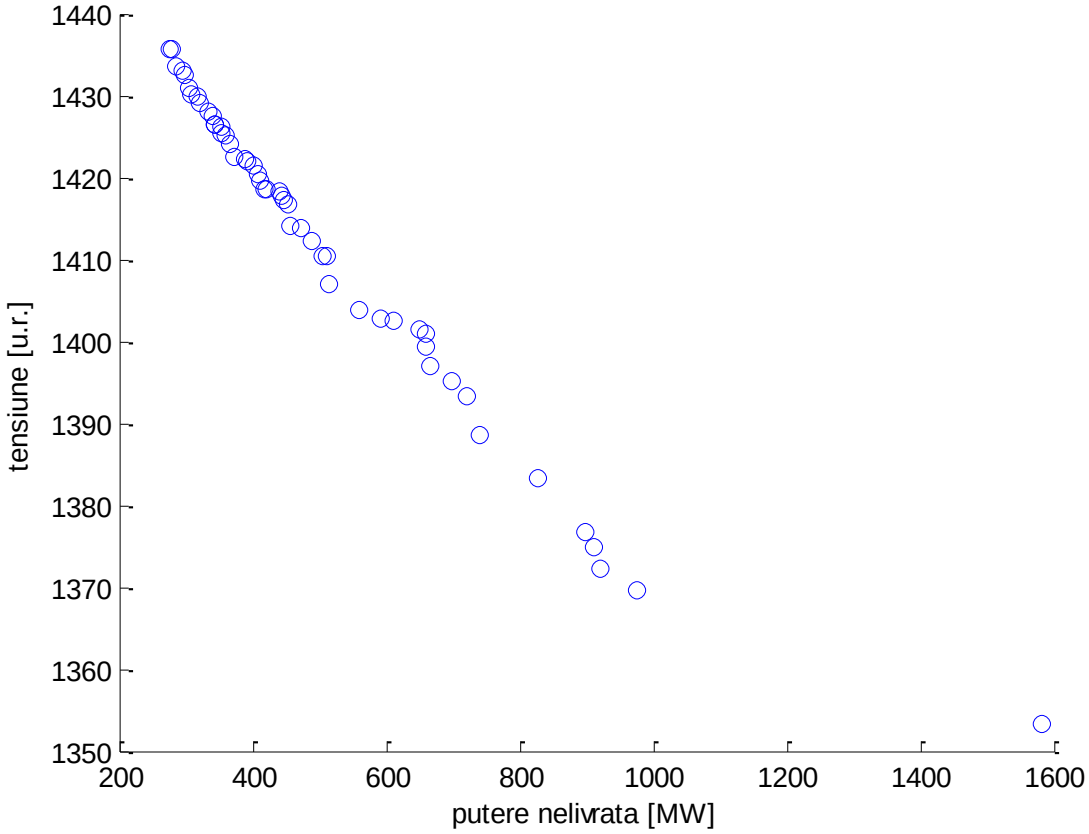
iteraţia 400

Evoluția unui front Pareto



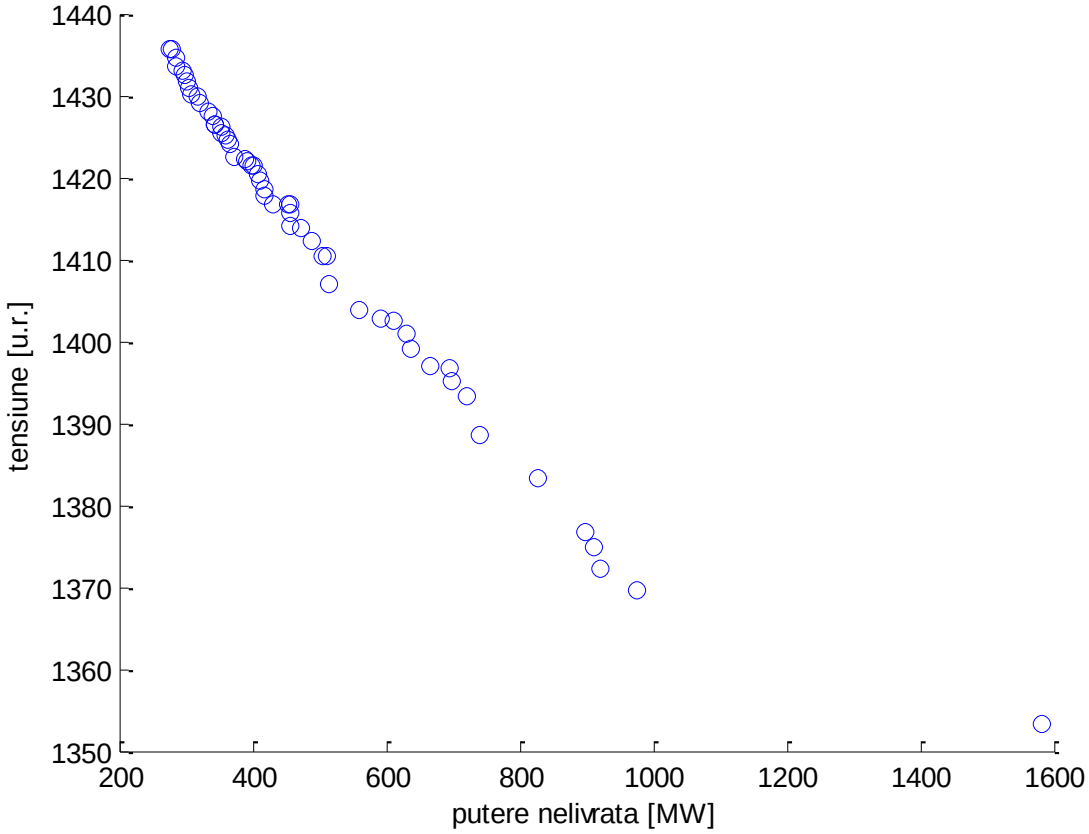
iteraţia 500

Evoluția unui front Pareto



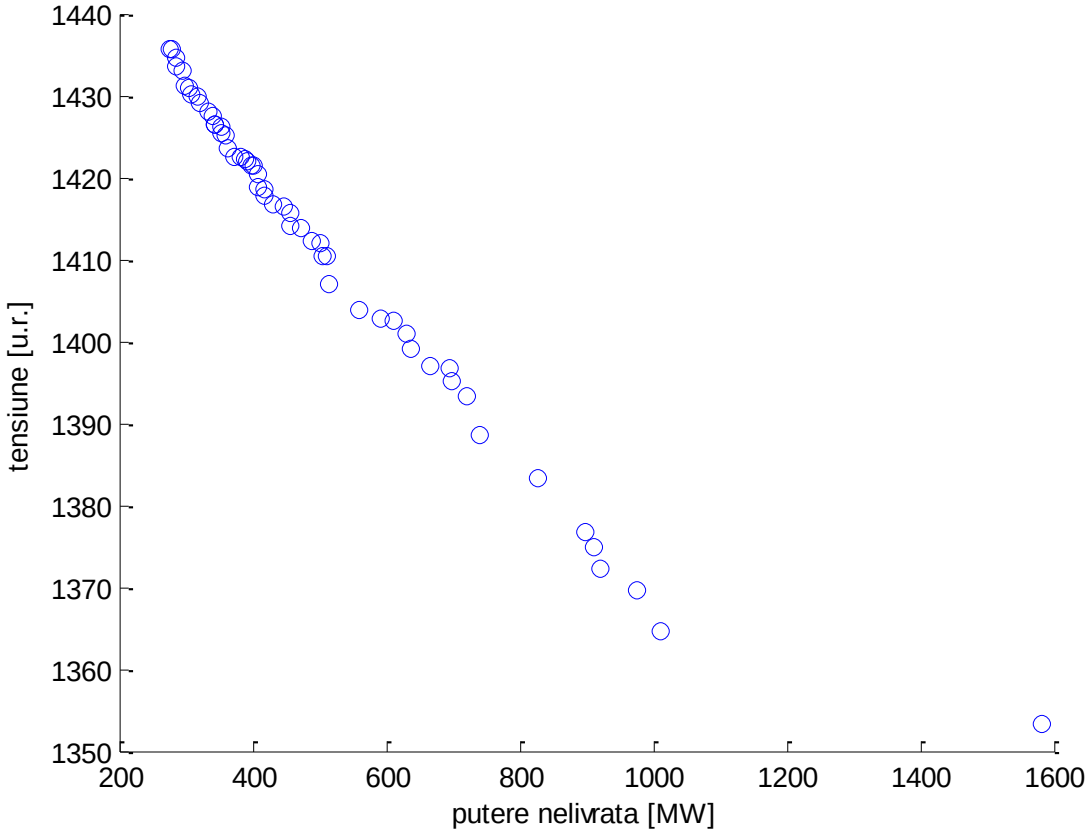
Iteraţia 600

Evoluția unui front Pareto



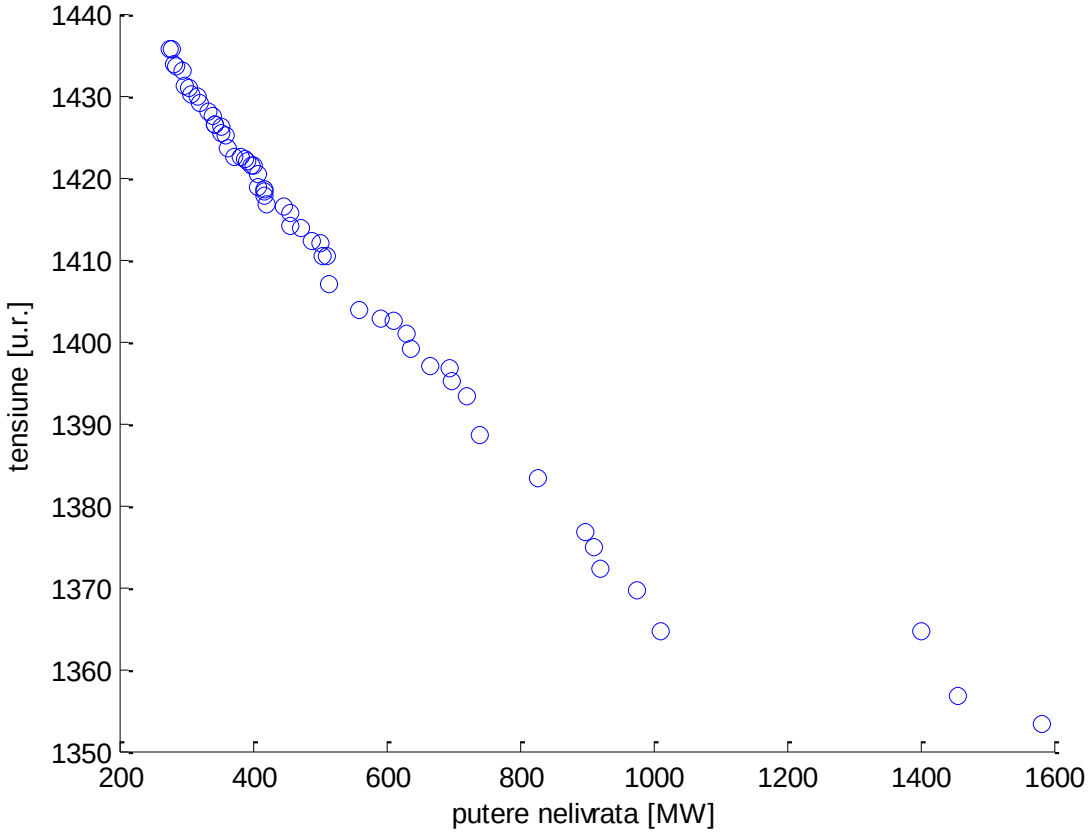
Iteraţia 700

Evoluția unui front Pareto



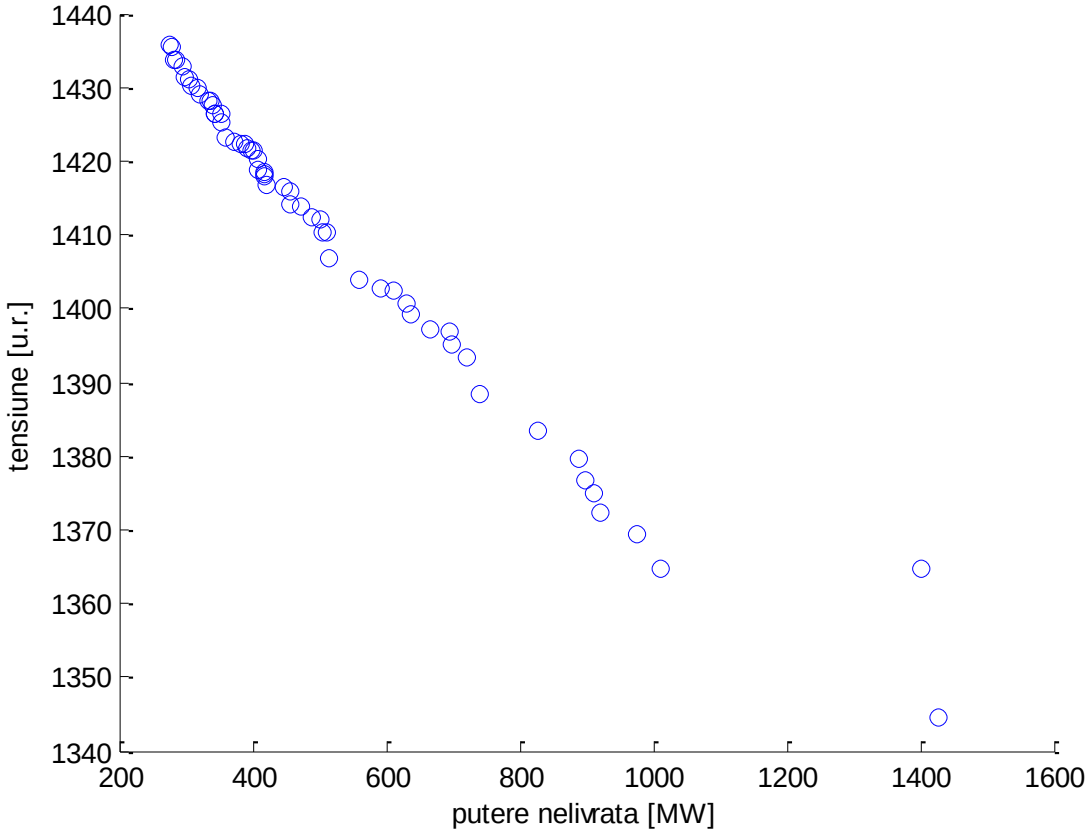
Iteraţia 800

Evoluția unui front Pareto



Iteraţia 900

Evoluția unui front Pareto



Iteraţia 1000

Exemplu de implementare a optimizării multiobiectiv la algoritmele genetice

- Funcţia de adaptare calculată pentru un cromozom AG poate include două obiective ponderate:

$$FA = pondere_1 \cdot FO_1 + pondere_2 \cdot FO_2$$

- Ponderile celor două obiective pot fi diferite.

va urma...

- Rețele neuronale

Vă mulțumesc pentru atenție !