Scalabilitatea algoritmilor metaeuristici

- Motivație
- Modele paralele ale calculului evolutiv
- Implementări distribuite ale algoritmilor evolutivi
- Coevoluție cooperativă

Motivație

Algoritmii metaeuristici bazati pe populatii necesită un volum mare de resurse:

- Spațiu memorie (întrucât operează cu populații)
- Timp execuție (întrucât necesită efectuarea unui număr mare de generații)

Operații costisitoare:

- Evaluarea elementelor populației
- Aplicarea operatorilor

Soluții:

- Optimizarea algoritmului (dezvoltarea de noi operatori)
- Optimizarea implementării (implementare paralelă/distribuită)

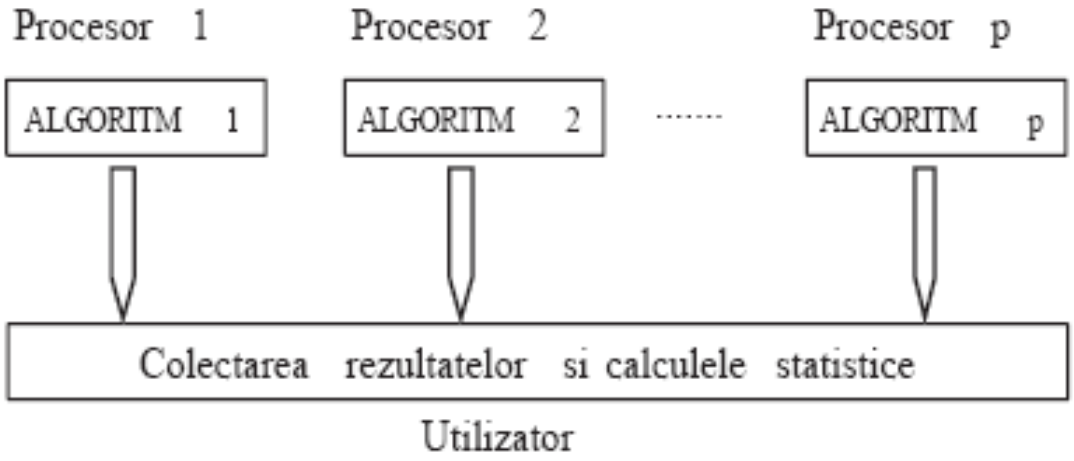
Modele paralele si distribuite

Paralelizarea se poate realiza la unul dintre nivelele:

- Algoritm -> modelul naiv de paralelizare
- Evaluarea elementelor -> modelul master-slave
- Populație -> modelul insular
- Element -> modelul celular

Modelul naiv

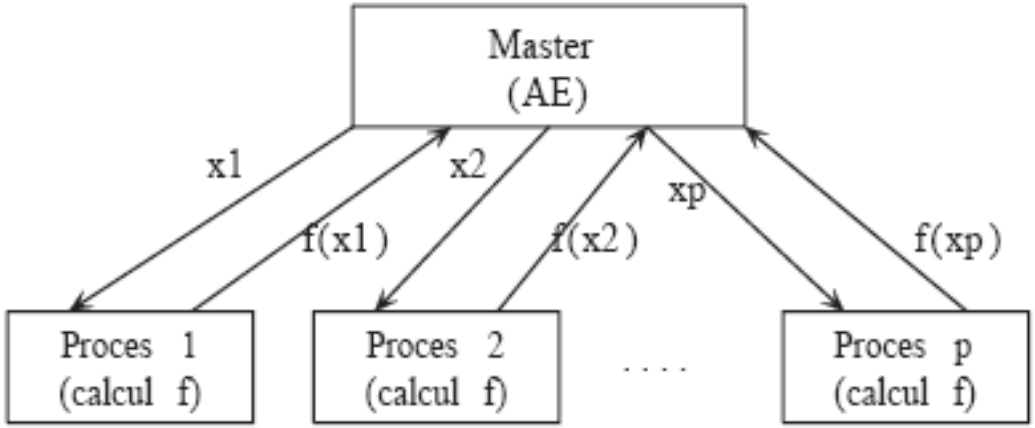
Se execută algoritmul simultan pe mai multe procesoare care nu comunică între ele



Este util în cazul în care se dorește efectuarea unor analize statistice (algoritmii fiind aleatori nu este suficientă o singură rulare a algoritmului) cu scopul evaluării algoritmului sau a identificarii valorilor potrivite pentru parametrii de control

Modelul master-slave

Procesul master execută algoritmul metaeuristic și distribuie evaluarea elementelor către procesele sclav



Modelul master-slave

Specific:

- Dacă volumul populației este mai mare decât numărul de procesoare atunci procesul master are sarcina de a repartiza evaluările către fiecare procesor

- Durata evaluării poate depinde nu doar de caracteristicile procesorului ci și de cele ale elementului de evaluat (de exemplu în cazul programării genetice) când evaluarea unor elemente necesită mai puțin timp iar a altor elemente mai mult timp

- In cazul algoritmilor generaționali (când trebuie evaluată întreaga populație înainte de a trece la generația următoare) apare problema timpilor de așteptare. Pentru a evita acest lucru se poate înlocui strategia sincronă (generațională) de actualizare a populației cu o strategie asincronă (steady-state)

Modelul master-slave

Sincronă

Inițializare populație

Evaluare populație

REPEAT

- Selecție părinți
- Generarea populației de urmași prin aplicarea operatorilor de variație
- Evaluarea populației de urmași
- Selecția supraviețuitorilor

UNTIL <condiție de oprire>

Asincronă

Inițializare populație

Evaluare populație

REPEAT

- Selecție părinți
- Generarea unui nou element
- Evaluare element
- Asimilare element în populație

UNTIL <condiție de oprire>

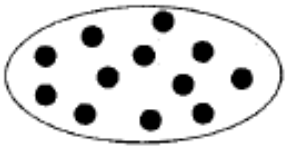
Modelul master-slave

- Este ușor de implementat
- Conduce la implementări eficiente doar dacă operația de evaluare este semnificativ mai costisitoare decât celelalte operații (pentru a se compensa costul indus de comunicarea intre procesoare)
- Comportarea algoritmului evolutiv (din perspectiva proprietăților de convergență) nu este modificată
- Poate fi implementat atât pe sisteme multiprocesor cu memorie partajată cât și pe sisteme cu memorie distribuită, inclusiv pe rețele de calculatoare

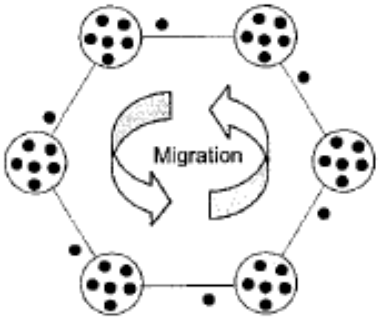
Structurarea populatiei

- Populațiile pot fi nestructurate (panmictice) sau structurate
- Structurarea populației influențeaza procesul de evoluție, unul dintre principalele efecte fiind stimularea diversității populației
- Structurarea populației poate fi cu granularitate:
  - Mare (model insular)
  - Mica (model celular)

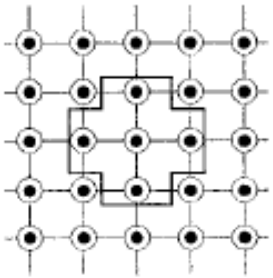
Alba, Tomassini; Parallelism and EAs, 2002



Model panmictic



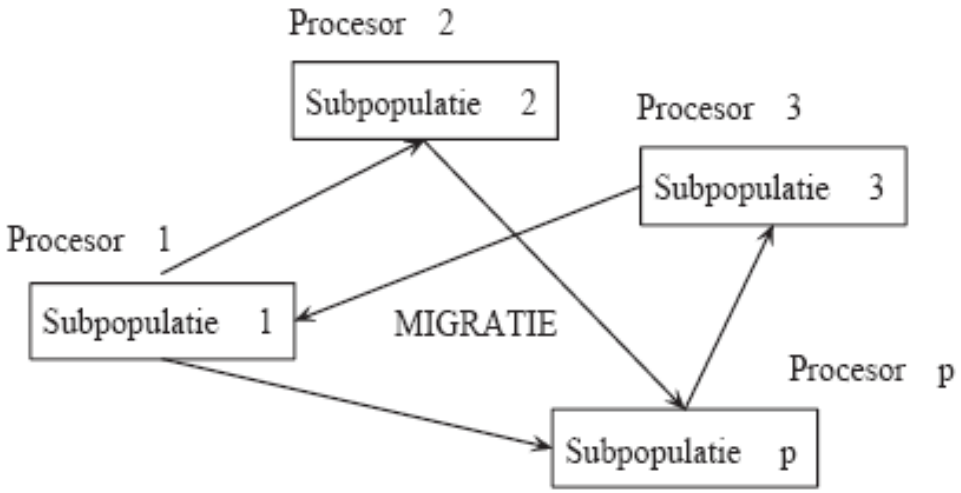
Model insular



Model celular

Modelul insular

- Se bazează pe divizarea populației în subpopulații (numite și “deme”) în care se aplică algoritmi identici sau diferiți și care comunică între ele printr-un proces de migrare



- Un procesor se poate ocupa de una sau mai multe subpopulații
- In fiecare subpopulație se aplică transformările specifice metaeuristicii pentru un anumit număr de generații după care este inițiat un proces de migrare

Modelul insular

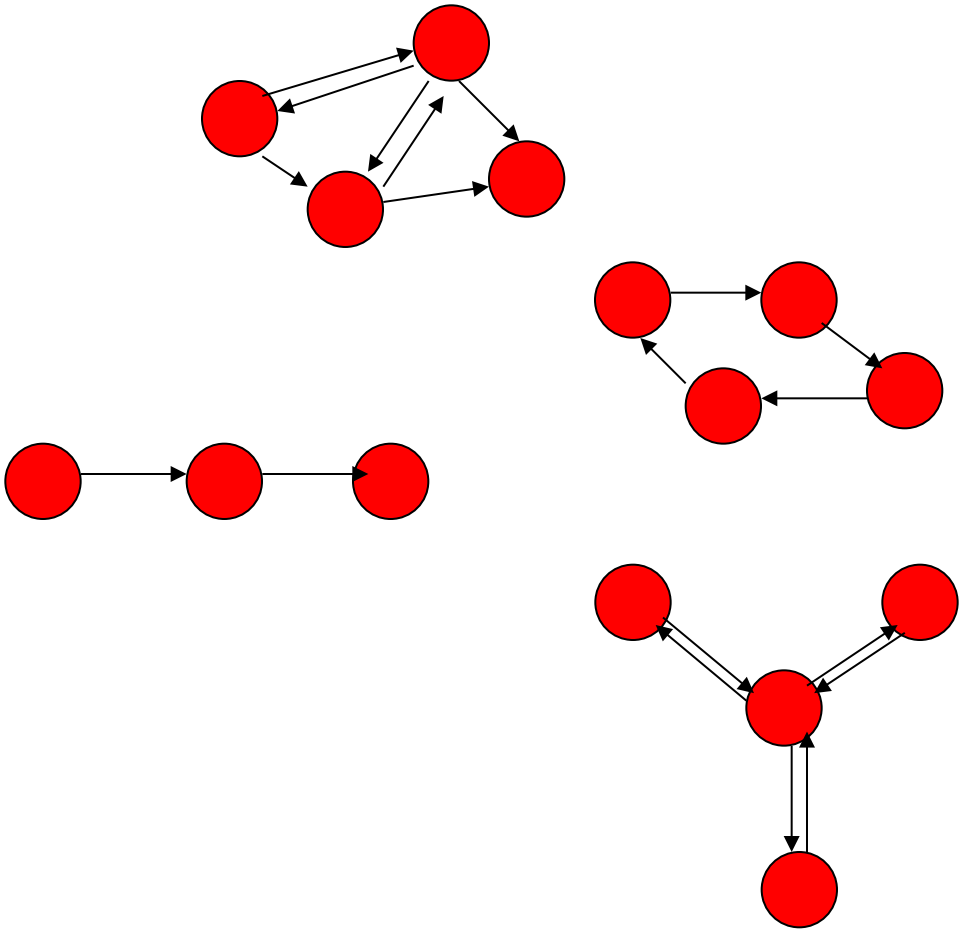
Procesul de comunicare (migrare) dintre (sub)populații este caracterizat de:

- Topologia de comunicare
- Strategia de comunicare
- Parametrii de control ai comunicării

Aceste elemente influențează semnificativ modul de comportare al algoritmului și eficiența acestuia

Modelul insular

- Topologie de comunicare
  - Topologie aleatoare
  - Topologie inel
  - Topologie liniară
  - Topologie stea

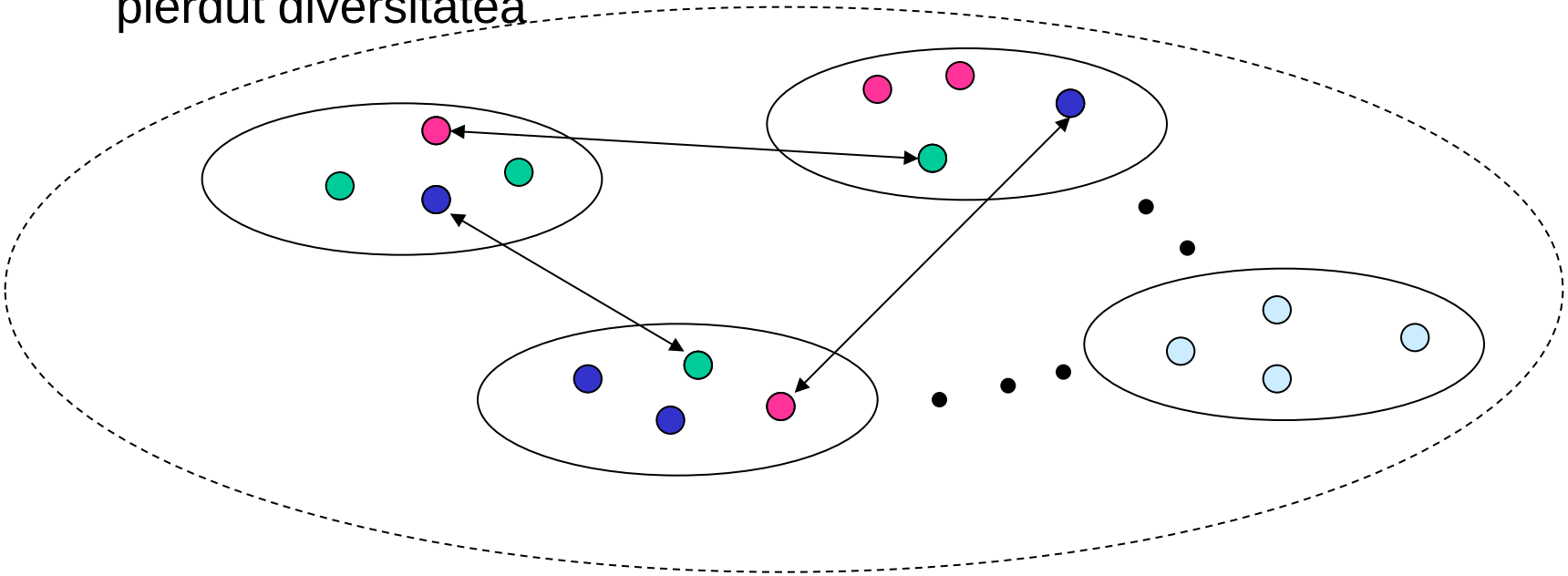


Modelul insular

- Strategie de comunicare
  - Migrare propriu-zisă: un element este transferat în altă populație din care este preluat un alt element în loc
  - Polenizare: o copie a unui element este transferată într-o populație destinație unde înlocuiește un alt element (care în felul acesta este eliminat complet din populație)
- Selecția emigrantului:
  - Aleatoare (fiecare element are o anumită probabilitate de selecție) – se aplică de obicei la migrare
  - Elitistă (se alege unul dintre cele mai bune elemente) – se aplică de obicei la polenizare
- Selecția elementului ce va fi înlocuit (în cazul polenizării)
  - Aleatoare - migrare
  - Elitistă (dintre cele mai slabe elemente) – polenizare

Modelul insular

- Exemplu simplu:
  - Interschimbare elemente
  - Distribuția elementelor la nivelul întregii populații rămâne neschimbată; se schimbă doar distribuția elementelor la nivelul subpopulațiilor; permite reactivarea subpopulațiilor care și-au pierdut diversitatea

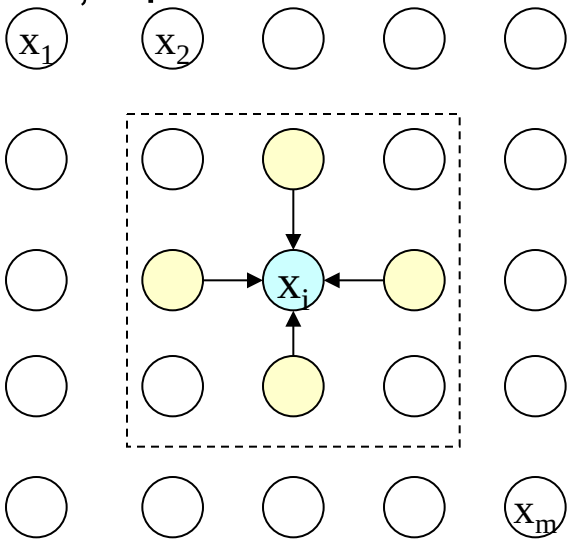


Modelul insular

- Parametri specifici:
  - Frecvența de migrare: determină momentul în care se activează procesul de migrare
    - Determinată de numărul de generații (exemplu: din 50 în 50 de generații)
    - Determinată de proprietățile populației (exemplu: când diversitatea populației a scăzut sub un anumit prag)
  - Probabilitate de migrare:
    - Cu cât este mai mare cu atât sunt mai multe elemente transferate între (sub)populații

Modelul celular

- Elementele populației sunt plasate în nodurile unei grile (cu structura toroidală) pe care este definită o relație de vecinătate
- Doar elementele vecine comunică și participă la aplicarea operatorilor evolutivi
- In implementarea paralelă, fiecărui element i se asociază un procesor (modelul este adecvat pentru execuția pe un supercalculator)



http://neo.lcc.uma.es/cEA-web/index.htm

Modelul celular

- Modelul celular poate fi utilizat și în contextul implementării secvențiale – conduce la o dinamică diferită a populației față de cea corespunzătoare populațiilor nestructurate

- Algoritmii evolutivi celulari sunt într-o oarecare măsură similari automatelor celulare sau rețelelor neuronale cu arhitectură celulară

- La fel ca și in cazul populatiilor nestructurate exista două variante de implementare:

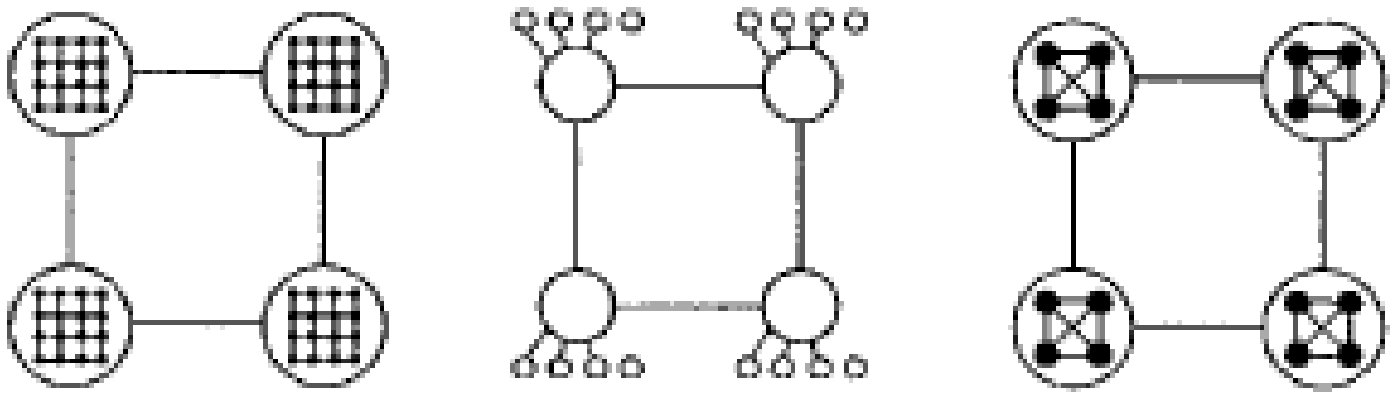
  - Sincronă: toate elementele populației sunt simultan înlocuite cu cele obținute prin încrucișare și mutație
  - Asincronă: elementele create prin încrucișare și mutație își înlocuiesc părinții imediat după ce au fost construite

Modelul celular

- Variante de evoluție asincronă:
  - Elementele sunt alese în manieră aleatoare (selecție uniformă fără revenire)
  - Celulele grilei sunt parcurse linie cu linie
  - Se construiește o permutare aleatoare de ordin m (m este numărul de elemente din populație) iar elementele sunt prelucrate în ordinea specificată de această permutare
- Varianta asincronă conduce la o convergență mai rapidă decât cea sincronă

Variante hibride

- Cele trei tipuri de modele de paralelizare pot fi hibridizate în diferite moduri
  - Insular+celular: Fiecare subpopulație conține un algoritm celular
  - Insular+MasterSlave: prelucrările din fiecare subpopulație (în special evaluarea funcției obiectiv) sunt executate în paralel
  - Insular+insular: împărțirea în subpopulații este realizată la două nivele (structură ierarhică)

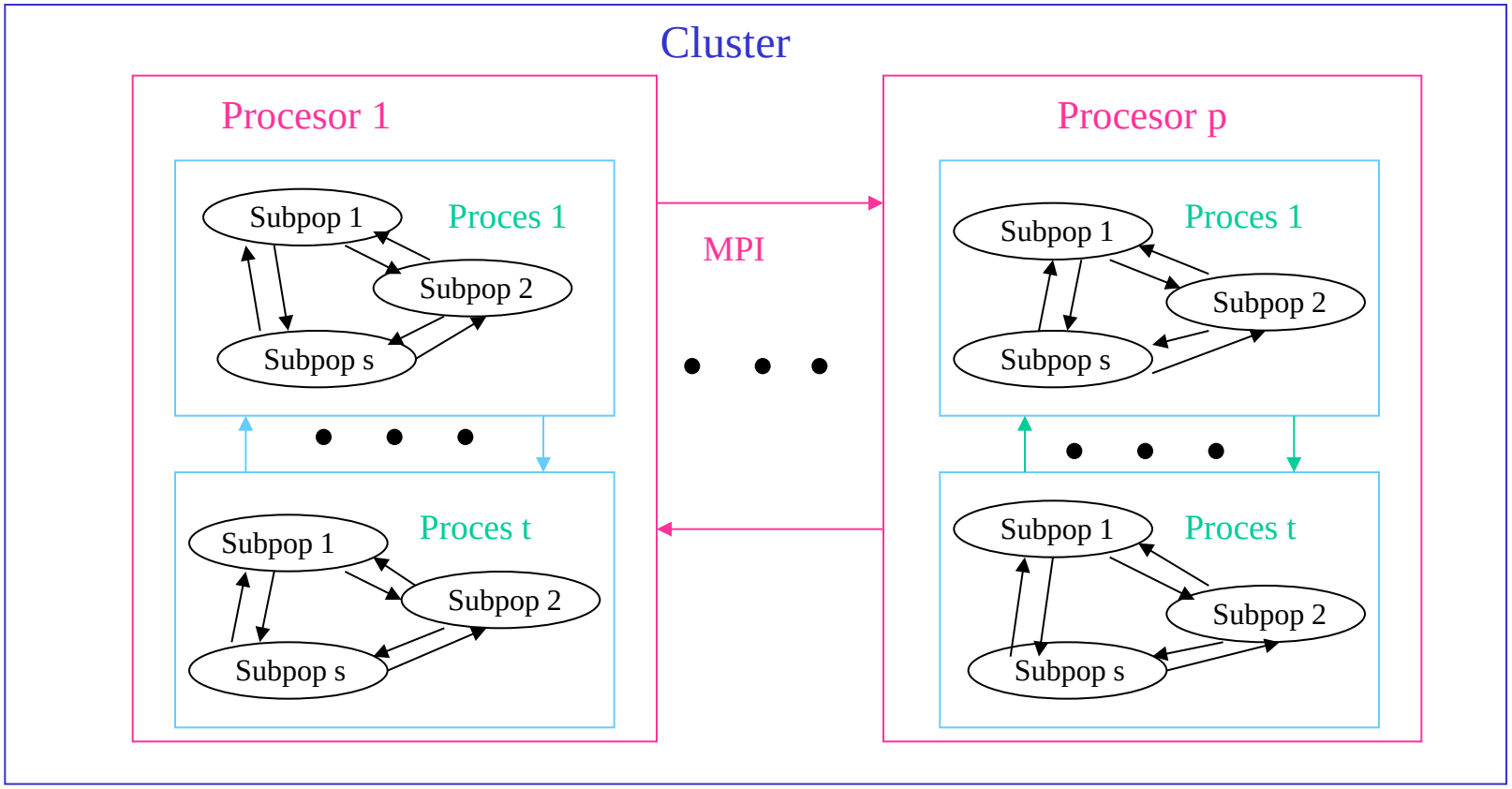


Implementare

- Mediul de calcul adecvat depinde de gradul de granularitate și de intensitatea comunicării intre componentele populatiei
  - Modelul master-slave poate fi implementat pe arhitecturi de tip cluster
  - Modelul insular poate fi implementat eficient pe arhitecturi de tip cluster sau chiar pe arhitecturi de tip grid dacă comunicarea între subpopulații este foarte rară
  - Modelul celular este eficient pe arhitecturi multi-procesor (întrucât necesită comunicare intensivă între nodurile de prelucrare)
- Instrumente software: MPI, OpenMP etc.

Implementare

- Exemplu de distribuire a prelucrărilor pe procese și procesoare în cazul modelului insular



Tendința curentă

Implementări pe arhitecturi GPU și hibride (CPU+GPU)

- Varianta master-slave
  - Algoritmul evolutiv este executat pe CPU
  - Evaluarea elementelor populației pe GPU

- Modele insulare
  - Modelul insular nu e foarte adecvat pentru implementare pe GPU
  - O posibilă variantă e cea în care
    - Populația inițială este generată pe CPU și stocată in GPU VRAM
    - Pentru fiecare subpopulație se execută un algoritm evolutiv pe GPU
    - Procesul de migrare este realizat prin intermediul GPU VRAM

Biblio: Arenas et al., GPU Parallel Computation in Bioinspired Algorithms. A review. - 2012

Tendința curentă

- Modele celulare
  - Structura bi-dimensională care definește interacțiunile dintre elementele populației este mapată pe structura GPU (fiecare element al populației este prelucrat de către un element de procesare din GPU)
  - Intreg algoritmul evolutiv este executat pe GPU (doar generarea de numere aleatoare este executată pe CPU – se pot genera la începutul algoritmului și stoca în memorie o serie de valori aleatoare pentru a fi utilizate de către algoritmul evolutiv). Obs: există și implementări în care valorile aleatoare sunt generate de către GPU
  - Există implementări în care toate prelucrările sunt efectuate pe GPU fără a folosi CPU pentru algoritmul evolutiv

Biblio: Arenas et al., GPU Parallel Computation in Bioinspired Algorithms. A review. - 2012

Coevoluție cooperativă

- Utilă în rezolvarea problemelor de dimensiuni mari (câteva sute sau mii de variabile)

- Ideea principală: se descompune problema în subprobleme de dimensiuni mai mici
  - O soluție candidat este constituită din mai multe componente
  - Populația poate fi interpretată ca fiind constituită din subpopulații corespunzătoare componentelor
  - Se aplică metaeuristica fiecăreia dintre subpopulații (coevoluție)
  - Evaluarea unei componente (element din subpopulația corespunzătoare) se poate face doar cunoscând valorile variabilelor aparținând celorlalte componente (cooperare)

- Principiul coevoluției este intâlnit în natura la specii care coabitează

Coevoluție cooperativă

Aspecte legate de implementare:

- Alegerea componentelor
  - Câte componente?
  - Cum se asignează variabilele acestor componente?

- Coevoluția componentelor
  - Cum se stabilește care e contextul de evaluare a unei componenta (valorile variabilelor din celelalte componente)?
  - Cât de mult se poate păstra același context fără a altera semnificativ performanța

Coevoluție cooperativă

Cel mai simplu caz:

- Se asignează fiecare variabilă unei componente = o problemă de dimensiune n este descompusă în n probleme uni-dimensionale

- Această abordare este potrivită pentru problemele separabile (de exemplu dacă f(x1,x2,...,xn)=f1(x1)+f2(x2)+...+fn(xn))

Dar nu funcționează in cazul în care funcția obiectiv este neseparabilă (de exemplu: f(x1,x2)=100(x1-x2)^2+(1-x1)^2) – intr-un astfel de caz calitatea relativă a unor variabile depinde de valorile celorlalte variabile)

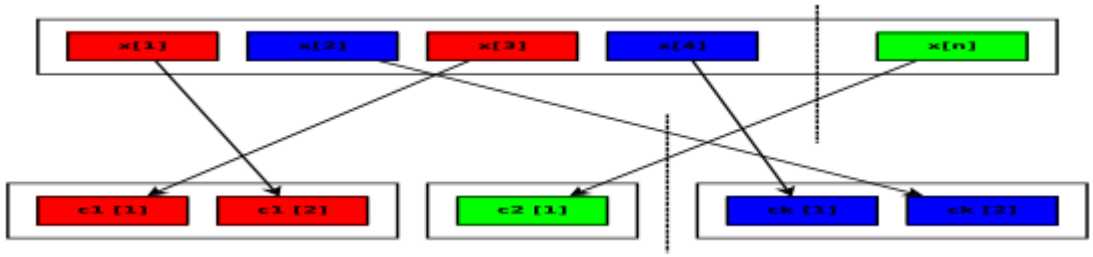
Coevoluție cooperativă

Cazul ideal:

- Fiecare componentă conține variabile care sunt inter-corelate iar in componente diferite sunt variabile necuplate sau slab cuplate

Varianta de compromis:

- Se (re)asignează (la fiecare iterație) variabilele aleator la componente
- Functioneaza acceptabil doar daca interactiunile se limiteaza la perechi de variabile



Coevoluție cooperativă

Differential grouping:

- Se estimeaza pentru fiecare variabila gradul de interactiune cu alte variabile
- Idee: doua variabile i si j se considera ca fiind in interactiune daca pentru un vector arbitrar x si doua valori perturbatoare d1 si d2 are loc

f(...,xi+di,...,xj+dj,...)-f(...,xi+di,...,xj,...) !=

f(...,xi,...,xj+dj,...)-f(...,xi,...,xj,...)

- Pentru fiecare variabila se estimeaza nr de alte variabile cu care interactioneaza, se sorteaza lista de variabile pe baza acestui numar si se realizeaza separarea in componente (sansa ca variabile care interactioneaza sa fie impreuna este mai mare)

[Omidvar et al., Cooperative Coevolution with Differential Grouping for Large Scale Optimization, 2013]

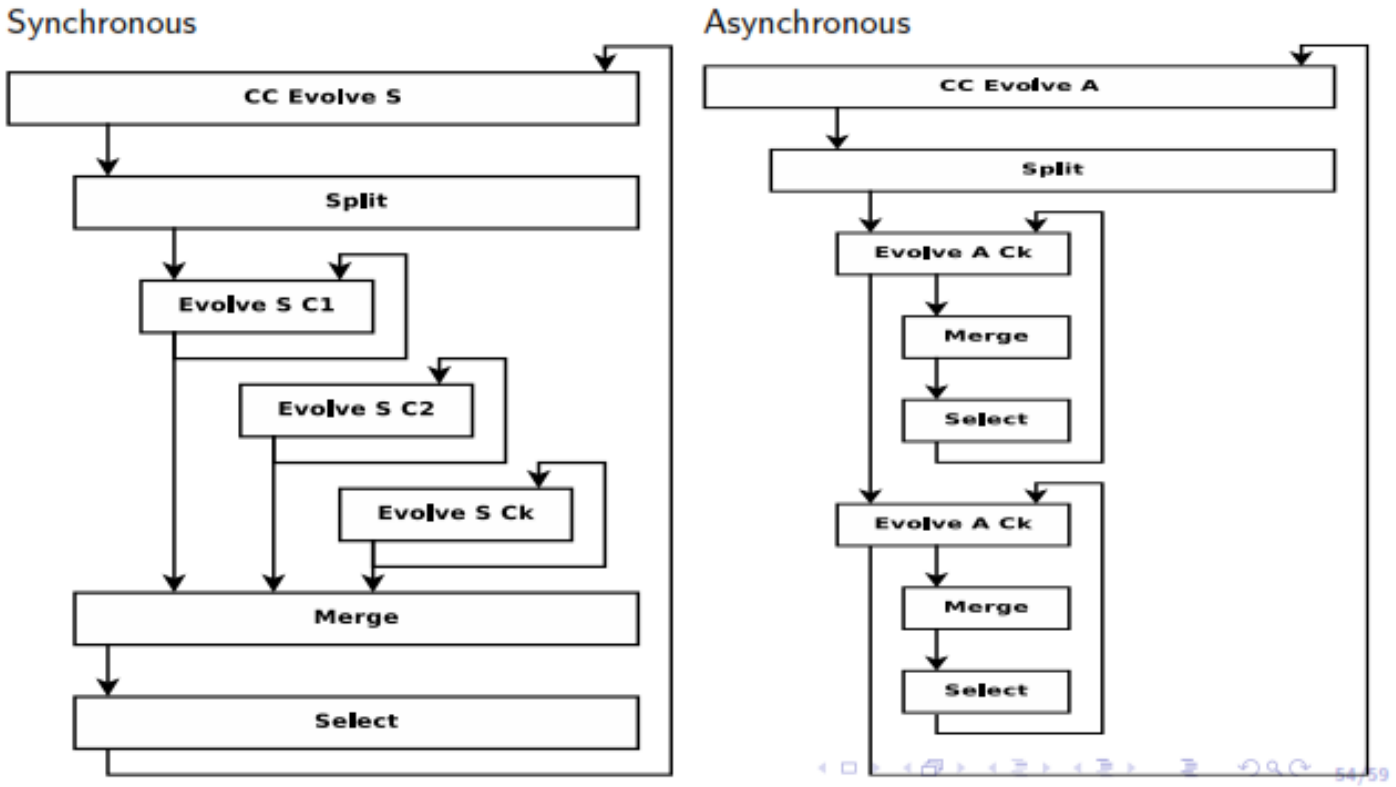
Coevoluție cooperativă

Alegerea contextului de evaluare:

- Pentru a evalua o componenta c(k) trebuie construita o solutie, (c(1),...,c(k),....,c(K)) prin definirea unui context de evaluare constituit din colaboratori provenind din subpopulatia corespunzatoare fiecareia dintre celelalte componente
- Un colaborator poate fi :
  - Cel mai bun element din subpopulatia corespunzatoare
  - Component corespondenta din cel mai bun element al populatiei globale
  - Component corespondenta dintr-un element aleator al populatiei globale

Coevoluție cooperativă

Variante de implementare:



Coevoluție cooperativă

Impactul coevolutiei cooperative asupra nr de evaluari de functii necesar pt a se atinge o anumita acuratete:

Grafic:
nfe(n)/nfe(100) versus n (dim problemei)

Algoritm: Differential Evolution (curs 8)