REFERAT 1

Stadiul actual în cercetarea sistemelor şi algoritmilor paraleli din recunoasterea formelor

Conducător ştiinţific

prof.univ.dr.ing. Ştefan Gheorghe Pentiuc

Doctorand

ing. Gabriel Anastasiu

Cuprins:

Cap.1 Metode actuale de recunoaştere a formelor şi tipuri de algoritmi utilizaţi în recunoaşterea formelor

Recunoaşterea formelor este o metodă de cercetare care se poate defini astfel: clasificarea datelor de intrare în clase identificabile prin intermediul determinării caracteristicilor sau atributelor semnificative ale datelor, relativ la un mediu cu detalii irelevante. Există sisteme care pot recunoaşte caractere, recunosc faţa umană, recunoaşterea amprentelor digitale, recunoaşterea vocii. Deci putem redefini recunoaşterea formelor ca fiind un proces de modelare, reprezentare, clasificare, prelucrare, stocare şi generare pe calculator a formelor.

Scopul recunoaşterii formelor constă în determinarea clasei din care face parte o colecţie de observabile. Metoda este deosebit de utilă atunci când abordarile directe sunt imposibile sau când inferenţele teoretice lipsesc.

Stabilirea numărului de clase în care se împart formele este o problemă particulară care depinde exclusiv de aplicaţiile concrete ale metodei.

Un sistem de recunoaştere a formelor trebuie să asigure, corect şi eficient observarea, transformarea, prelucrarea preliminară (selectarea) şi clasificarea eşantionului de date.

Elementele esenţiale ale unui sistem general de recunoaşterea formelor sunt următoarele: translatorul, selectorul de caracteristici (care realizează o prelucrare preliminară) numit şi preprocesor, sau extractor de caracteristici şi clasificatorul (Fig. 1). Deşi aceste 3 subunităţi sunt interdependente, în cele ce urmează le vom prezenta separat.

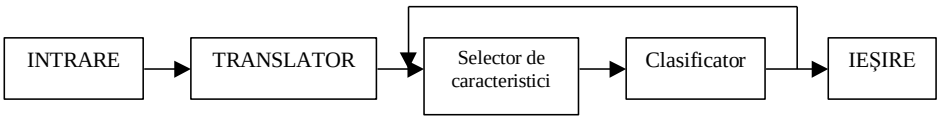


Figura.1 Schema bloc sistem recunoaştere forme

Unitatea de intrare o reprezintă de regulă un senzor optic care poate fi de tip High Definition şi care asigură interacţiunea cu subiectul. De sensibilitatea acestui dispozitiv depinde modul de prelucrare a imaginii şi recunoaşterea formei.

Translatorul este de regulă partea cea mai importantă a acestui sistem pentru că realizează transformarea imaginii achiziţionate prin unitatea de intrare pe baza unui algoritm matematic precis în semnal care apoi va fi prelucrat de către sistemul de calcul. De regulă în translarea informaţiei se folosesc diverse tipuri de transformate Fourier. Vectorii de formă dezvoltaţi de translator constituie mărimile de intrare pentru selectorul de caracteristici.

Translatorul transformă şi transferă informaţiile din lumea reală în spaţiul formelor într-o formă compatibilă cu modul de reprezentare din calculatoarele electronice. În

consecinţă datele primare, rezultat al observaţiei sunt transformate într-un şir de mărimi scalare care formează vectorul de formă *n*-dimensional. Fiecare componentă x_i a vectorului de formă *X* reprezintă o cantitate fizică măsurabilă; este foarte important ca ea să surprindă esenţa datelor primare.

Modul de implementare al translatorului depinde exclusiv de natura datelor primare. Dacă acestea sunt constituite dintr-o succesiune de valori măsurate la intervale de timp, cum sunt traseele EKG, atunci sunt necesare procedee de eşantionare în timp, pe când dacă ele sunt funcţie de frecvenţă, cum sunt de exemplu spectrele în infraroşu ale compuşilor chimici, atunci trebuie dezvoltate procedeele de eşantionare a frecvenţei (respectiv numerelor de undă). În cazul imaginilor sunt luate în considerare suprafeţele mai luminoase sau mai întunecate, muchiile sau formele geometrice. Aceasta este o problemă ceva mai complicată şi, de aceea , au fost propuse o serie de metode pentru reducerea complexităţii imaginilor la un şir de măsurători.

Selectorul de caracteristici

Scopul selectorului de caracteristici constă în prelucrarea vectorilor de formă în aşa fel încât procedeul de clasificare să fie optimizat.

Selectorul de caracteristici care mai este denumit şi extractor de caracteristici sau preprocesor primeşte ca mărimi de intrare vectorii de formă produşi de translator şi operează asupra lor transformându-i pentru a elimina sau pentru a reduce cantitatea de informaţie irelevantă sau ambiguă menţinând în vectori suficientă informaţie pentru a putea face deosebirea între diferitele clase de forme şi pentru a descoperi invarianţele dintre formele aceleiaşi clase.

Pentru realizarea acestor deziderate au fost propuse şi utilizate o mare varietate de metode.

Una dintre cele mai simple metode pentru prelucrarea vectorilor de formă constă în normarea acestora. O astfel de normare implică egalarea sumei componentelor fiecărui vector de formă în fapt suma pătratelor componentelor lor, cu o constantă arbitrară convenabil aleasă. Un alt procedeu, este cel care utilizează matricea de covarianţă duce, în final, la o ecuaţie matricială din care se obţin vectorii proprii şi valorile proprii .Procedeul este cunoscut ca analiza componentelor principale sau analiza Karhuneu-Loeve.

Pentru prelucrarea vectorilor de formă şi selectarea celor mai reprezentative caracteristici au fost utilizate şi o serie de transformări mult mai complexe, cum ar fi transformata Fourier.

Pentru identificarea caracteristicilor mai importante au fost utilizate forme model sau prototip, s-au dezvoltat şi implementat tehnici interactiv, implicând reprezentări grafice şi rutine speciale de comparare, s-au calculat parametrii statistici, cum sunt momentele sau histogramele direct din forme.

Clasificatorul de caracteristici

Clasificatorul de caracteristici are următoarea sarcină: având dată o mulţime de vectori de formă prelucraţi corespunzător, numită set de formare, se pune problema determinării unei funcţii de decizie *f(X)* astfel încât dacă:

$f(X) > 0$ atunci X aparţine clasei 1
$f(X) >= 0$ atunci X aparţine clasei 2 (1.1)

Această etapă în care este determinată funcţia de decizie *f(X)* este cunoscută sub numele de fază de formare (formarea), de adaptare sau uneori de învăţare. Scopul urmărit este minimalizarea probabilităţii de eroare în procesul de clasificare.

Conceptul de clasificare a formelor poate fi înţeles ca o partiţionare a spaţiului formelor, $\Omega_x = \{X\}$ prin atribuirea fiecărui vector X sau punct $X (x_1, ...,x_n)$ la o clasă de forme corespunzătoare în regiuni reciproc exclusive, fiecare regiune corespunzând unei clase de forme particulară. Din punct de vedere matematic problema clasificării poate fi formulată sub forma funcţiilor de decizie discriminate.

Fie $\omega_1, \omega_1, ... \omega_p$ cele p clase distincte posibile care urmează a fi recunoscute cu

$$\Omega_X = \omega_1 \cup \omega_2 \cup \ldots . \cup \omega_p,$$
$$\omega_1 \cap \omega_2 \cap \ldots\ldots \cap \omega_p = F_d \qquad (1.2)$$

şi fie $X=|x_i|_{i=1,n}$ vectorul de formă, x_i reprezentând a i-a caracteristică reprezentativă. Atunci funcţia de decizie discriminant $f(X)=D_j(X)$ asociată clasei de forme ω_j, $j=1,...,p$, astfel încât dacă forma de intrare reprezentată prin vectorul X, respectiv punctul X, este în clasa ω_i, fapt pe care-l vom nota prin $X \sim \omega_i$, valoarea lui $D_i(X)$ trebuie să fie cea mai mare, adică pentru toţi $X \sim \omega_i$ vom avea satisfăcută relaţia:

$$D_i(X) > D_j(X), \qquad i, j = 1,\ldots,p. \qquad (1.3)$$

În felul acesta, în spaţiul formelor ω_x frontiera partiţiei, denumită limita de decizie, dintre regiunile corespunzând claselor ω_i şi respectiv, ω_j, poate fi definită prin următoarea relaţie:

$$F_d = D_i(X) - D_j(X) = 0 \qquad (1.4)$$

În figura 2. este reprezentat modelul unui clasificator care utilizează funcţiile discriminant. Forma de intrare este analizată conform relaţiei (1.4), clasificatorul furnizând drept ieşire indicele k aparţine $\{1,2,\ldots,p\}$ corespunzător clasei ω_k din care face parte forma respectivă X.

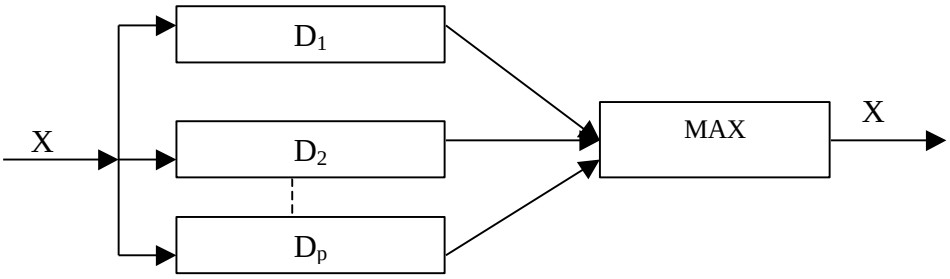


Figura 2 Schema bloc clasificator discriminant

Deci, după cum se poate constata din cele expuse până în acest moment recunoaşterea formelor constă din următoarele două aspecte importante:

1. Extragerea caracteristicilor esenţiale pentru procesul particular de clasificare. În mod obişnuit decizia care se ia în această etapă este relativ subiectivă şi depinde de considerente practice cum ar fi: accesibilitatea execuţiei măsurătorilor, costul acestora, etc. Criteriile care stau la baza procesului de selectare a caracteristicilor şi de ordonare a acestora se bazează fie pe importanţa lor în caracterizarea formelor, fie pe contribuţiile pe care le aduc la performanţele recunoaşterii.
2. Clasificarea propriu-zisă, adică luarea deciziei privind apartenenţa formelor la o clasă. Metodele matematice folosite în acest scop sunt adeseori denumite clasificatori. În figura figura 3. prezentăm schema bloc a unui sistem de recunoaştere a formelor.

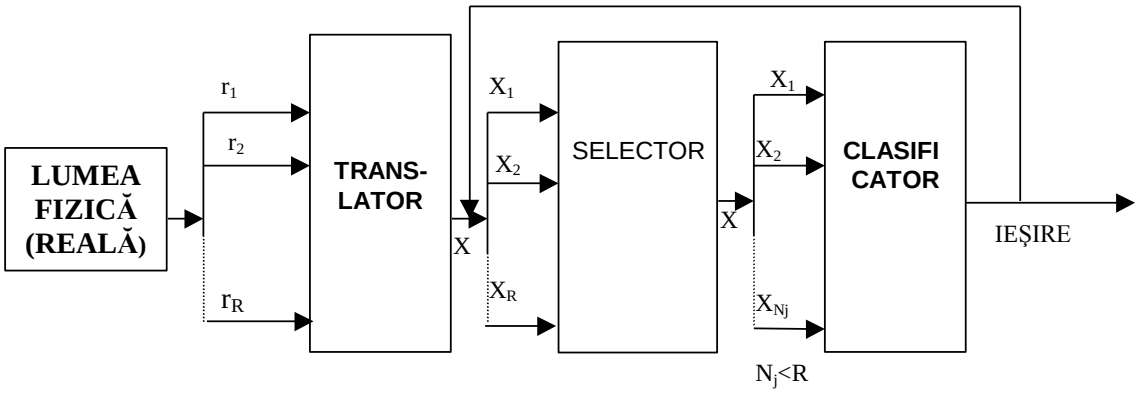


Figura 3 Schema bloc sistem de recunoaştere forme

O metodă de recunoaştere a formelor este metoda de recunoaştere controlată. Această metodă presupune existenţa unui set de forme a căror apartenenţă la clasă este cunoscută. Acest set este împărţit în două părţi: **setul de formare** utilizat pentru a dezvolta un clasificator care să recunoască cât mai bine apartenenţa formelor din set la clasele corespunzătoare şi setul de **predicţie** pe care clasificatorul format este testat. Clasificatorul astfel dezvoltat este utilizat în continuare pentru stabilirea apartenenţei unei forme necunoscute la o clasă.

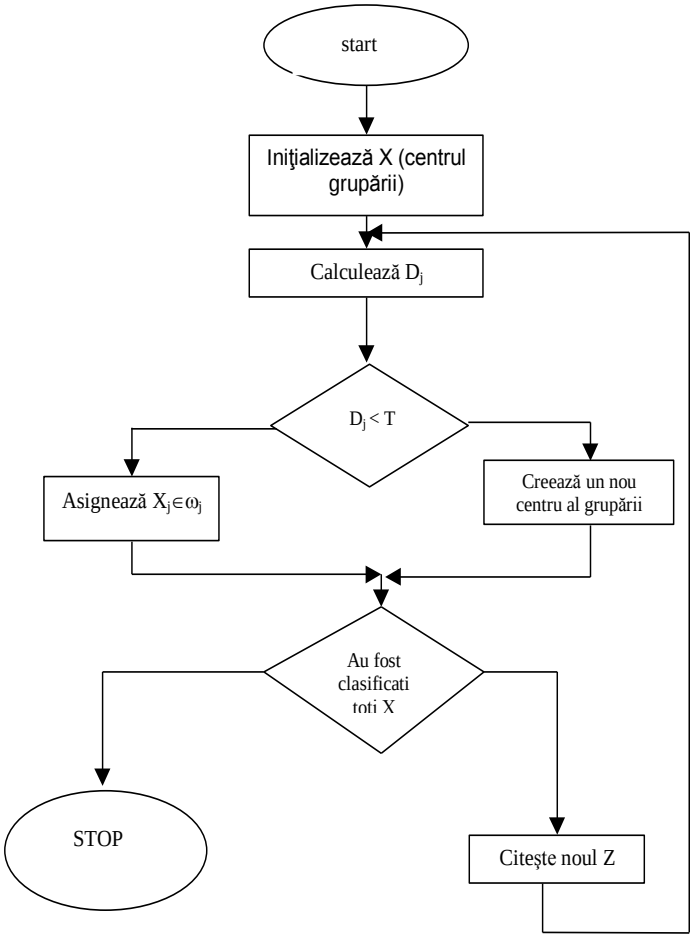
O altă metodă cunoscută este metoda de recunoaştere necontrolată. Metoda dezvoltă algoritmi care permit în cursul execuţiei acestora construirea claselor pe măsură ce formele

analizate sunt luate în considerare. Aceste metode sunt cel mai des utilizate şi constau într+un set de algoritmi care asigură împărţirea spaţiului formelor în clase şi grupe de forme.

Conceptul de grupare poate fi înţeles cel mai bine prin prezentarea celui mai simplu algoritm de grupare ,denumit algoritm de tip prag. Acest algoritm presupune existenţa în spaţiul formelor a unui set de forme şi stabilirea iniţială a unei distanţe minime,numită distanţa de prag, dintre două forme. Dacă distanţa dintre două forme este mai mică decât distanţa de prag, cele două forme fac parte din aceeaşi clasă. Acest algoritm este prezentat în figura 4 [2].

Studiind acest algoritm pot fi determinate o serie de caracteristici ale tehnicilor de grupare.

1. **Alegerea centrelor claselor (grupărilor).**
Modul de alegere afectează viteza de clasificare ca şi numărul de grupe sau clase care rezultă în urma executării procedurii de clasificare. Din acest motiv se recurge, de obicei, la calculul continuu a unui centru al clasei pe măsură ce la acesta se atribuie noi forme. În acest caz, centrul grupării poate să nu corespundă cu o forma existentă.



:

Figura 4 Algoritm alegere clase

2. Alegerea criteriului de clasificare.

În cazul exemplului dat, criteriul de clasificare este o distanţă. Se observă că valoarea lui *T* afectează rezoluţia procesului de clasificare. Dacă *T* este prea mare, două sau mai multe clase distincte pot fi grupate în una singură. În cazul în care *T* este prea mic, o grupare poate fi împărţită în mod artificial în câteva grupe. Pentru determinarea valori lui T se ţine cont de efectul pe care-l va avea această valoare asupra numărului de grupări. În cazul general se utilizează criteriile de similaritate şi nesimilaritate prin care se asigură apartenenţa unei forme la o clasă. Acestea pot fi distanţe sau alţi parametri.

Mai există o a doua posibilitate de descriere a algoritmului dată în figura 5 [2]:

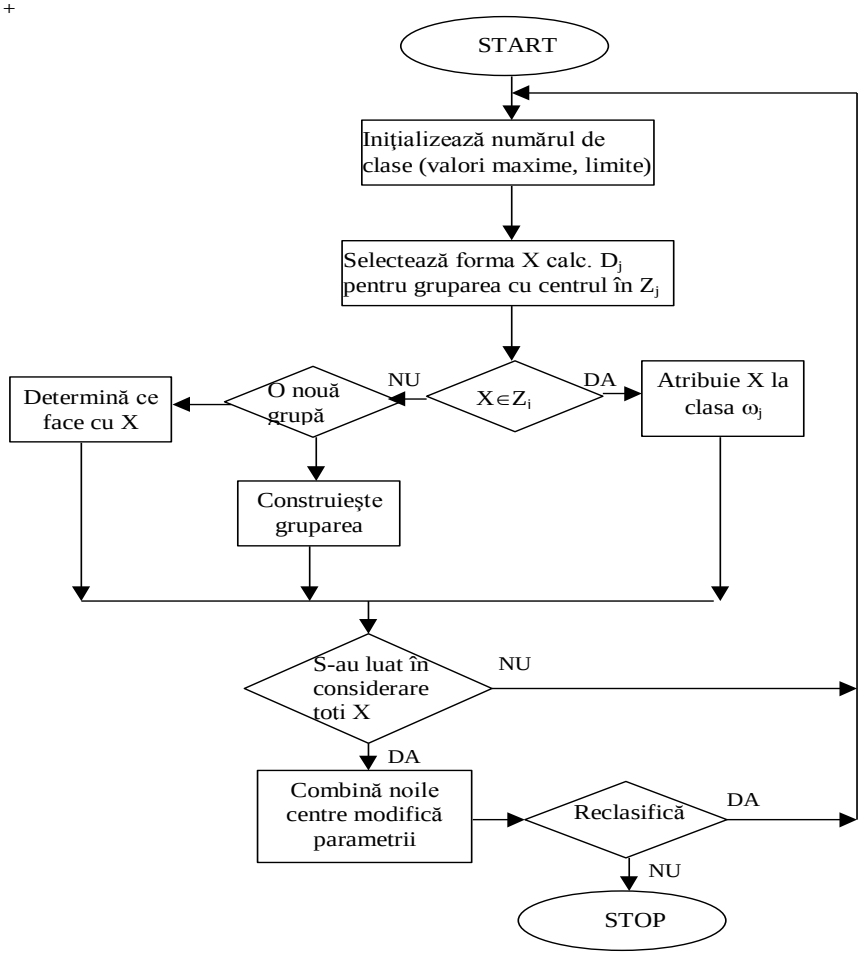


Figura 5 Algoritm selecţie clase modificat

1.2. Tipuri de algoritmi utilizaţi în recunoaşterea formelor

În acest capitol voi încerca să fac o scurtă trecere în revistă a principalilor algoritmi utilizaţi în recunoaşterea formelor [2][5].

Există la ora actuală câteva tipuri de algoritmi care sunt utilizaţi pentru procesele de recunoaşterea a formelor:

1. Algoritmul de tip grupare care se împarte în :

a. Algoritm de tip prag

b.Algoritm de tip grupare standard

2. Algoritmi de clasificare iterativă împărţiţi în:

a. Algoritmi n-medii

3. Algoritmi ISODATA

4. Algoritmi genetici şi evolutivi

1.2.1 Algoritmii de tip grupare

Aşa cum am exemplificat mai sus există două subtipuri de algoritmi tip grupare [2]:

a. Algoritm de tip prag

b. Algoritm de tip grupare standard

Aceşti algoritmi au fost descrişi pe larg în subcapitolul anterior.

1.2.2 Algoritmi de tip n-medii [2]

Aceşti algoritmi sunt contruiţi respectând următorii paşi:

a. Se alege o partiţie iniţială $\underline{p}^0 = \{\underline{A}_1, \underline{A}_2, \ldots, \underline{A}_n\}$ a lui $\underline{X}$.

b. Se calculează prototipurile acestei partiţii cu formula :

$$\underline{L}_i = \frac{\sum_{j=1}^{p} A_{ij}\, \underline{x}^y}{\sum_{j=1}^{p} A_{ij}} = \frac{1}{p_i} \sum_{\underline{x} \in A_i} \underline{x}$$

c. Se calculează noua partiţie după regula

$\underline{x} \to \underline{A}_i$ dacă $\left\| \underline{x}^j - \underline{L}_i \right\| < \left\| \underline{x}^j - \underline{L}_k \right\|, \forall k \neq i$

d. Dacă noua partiţie coincide cu precedenta, atunci STOP. În caz contrar se merge la P2.

Un posibil rezultat de implementare a unui astfel de algoritm se poate vedea în figura 6:

Obiect	1	2	3
1	1	0	0
2	1	0	0
3	1	0	0
4	0	1	0
5	0	1	0
6	0	0	1

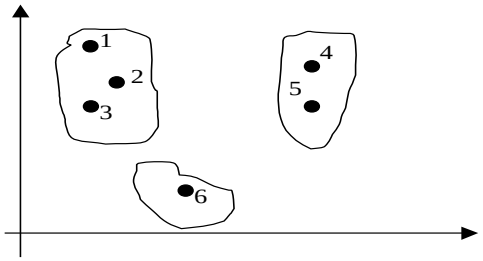


Figura 6 [2] Rezultat implementare algoritm n-medii

Referitor la datele din figura 6 putem trage câteva concluzii:

A) În loc de a alege o partiţie iniţială putem porni de la o alegere arbitrară a *n* centrii, care pot fi *n* puncte din mulţimea *X* a datelor.
B) Rezultatele algoritmului *n*-medii depind de numărul de clase considerate, de alegerea iniţială a partiţiei sau a centrilor şi de propietăţile geometrice ale datelor. Când datele conţin grupări caracteristice, relativ îndepărtate unele de altele, deci sunt constituite din clusteri compacţi şi bine separaţi, rezulatele algoritmului sunt bune. Determinarea numărului optim de clusteri prezenţi în mulţimea datelor constituie aşa numita problemă a validităţii clusterilor. Problema nu poate fi rezolvată în cadrul acestui algoritm.Putem ca prin experimentări succesive să determinăm valoarea care dă cea mai bună concordanţă cu datele iniţiale.
C) O altă problemă dificilă apare când datele conţin clase ce prezintă diferenţe mari ale numărului de puncte. Putem considera cazul când datele constau din doi clusteri inegali ca populaţie şi extindere. Există posibilitatea ca unele puncte aflate la periferia clusterului mare să fie mai apropiate de centrul clusterului mic. Funcţia criteriu considerată va favoriza o partiţie ce împarte clusterul mare, faţă de una ce menţine integritatea acestuia. Dacă se consideră *n=2*, atunci clusterul mai mic va capta unele din punctele periferice ale clusterului mare. Considerarea unei astfel de distanţe adaptive presupune în general ca algoritmul să fie aplicat de două ori.
D) Alte dificultăţi sunt legate de existenţa punctelor izolate (considerate zgomote) şi a podurilor între clustere.

Algoritmi ISODATA

Algoritmul ISODATA [2] (Iterativ Self-Organizing Data Analyst Techniques) este similar cu algoritmul *n-medii*. Această asemănare a făcut ca algoritmul *n-medii* să fie deseori prezentat sub numele de ISODATA. Asemănarea constă în modul iterativ de calcul a centrilor. Deosebirea este dată de natura euristică a algoritmului ISODATA. Algoritmul reprezintă un bun exemplu referitor la avantajele şi inconvenientele unor astfel de metode care necesită definirea unor parametrii ce trebuie ajustaţi prin încercări succesive.

Construcţia unui astfel de algoritm necesită parcurgerea următorilor paşi:

1. Se specifică valorile următorilor parametri:

   n = numărul de clase dorite;

   θ_p = numărul minim de elemente dintr-o clasă;

   θ_s = parametrul de dispersie standard;

   θ_c = distanţa maximă pentru fuzionare;

   L = numărul maxim de perechi de centre ce pot fuziona la un moment dat;

   I = numărul maxim de iteraţii permise.

2. Se distribuie *p* perechi din $\underline{X}$ în clasele determinate de centri, după regula:

$$\underline{x} \in \underline{Ai} \; dacă \; d\left(\underline{x}, \underline{m}^i\right) = \min\left\{d\left(\underline{x}, \underline{m}^j\right), \; j = \overline{1,k}\right\}$$

3. Se elimină clasele având mai puţin de θ_p elemente şi se repartizează obiectele în celelalte clase. Se micşorează *k*.

4. Se recalculează centrii claselor după formula

$$\underline{m}^i = \frac{1}{p_i} \sum_{\underline{x} \in \underline{Ai}} \underline{x}, \; i = \overline{1,k}$$

5. Se calculează diametrul mediu al fiecărei clase

$$D_i = \frac{1}{p_i} \sum_{\underline{x} \in \underline{Ai}} d\left(\underline{x}, \underline{m}^i\right), \; i = \overline{1,k}$$

6. Se calculează distanţa medie la care se află obiectele faţă de centrii claselor

$$D = \frac{1}{p} \sum_{i=1}^{k} \sum_{\underline{x} \in \underline{A_i}} d\left(\underline{x}, \underline{m}^i\right) = \frac{1}{p} \sum_{i=1}^{k} p_i D_i$$

7. 1) Dacă aceasta este ultima iteraţie atunci se pune θ_c şi se merge la 11

   2) Dacă $k \le \frac{n}{2}$ se merge la 8

   3) Dacă numărul iteraţiei este par sau dacă $k \ge 2n$ se merge la 11

8. Se calculează dispersia clasei $\underline{A}_i$

$$\underline{\sigma}_i^2 = \frac{1}{p_i} \sum_{\underline{x} \in \underline{A}_i} \left(\underline{x} - \underline{m}^i \right)^2, \; i = \overline{1,k}$$

Componenta σ_{ij} a lui σ_i reprezintă dispersia datelor în direcţia j.

9. Pentru fiecare clasă se determină componenta maximă a dispersiei

$$\sigma_{ij_m} = \max_j \sigma_{ij} \;\; i = \overline{1,k}$$

10. Se consideră succesiv toate clasele. Dacă există o clasă $\underline{A}_i$ astfel încât

$$\sigma_{ij_m} > \theta_s$$

şi

$$D_i > D \;, \;\; p_i > 2\left(\theta_p + 1\right)$$

sau

$$k < \frac{n}{2}$$

atunci se împarte clasa $\underline{A}_i$ în două clase şi se calculează centrii acestora

$$\underline{m}^{i+} \text{ şi } \underline{m}^{i-}$$

se şterge $\underline{m}^i$ şi se pune $k := k+1$. Centrul $\underline{m}^{i+}$ se obţine adăugând la componenta $m^i_{j_m}$ (care corespunde componentei maxime ale lui $\underline{m}^i$) o cantitate $\sigma_{ij_m} \cdot a$ (unde $0 < a \le 1$). Centrul $\underline{m}^{i-}$ se obţine scăzând această cantitate din $m^i_{j_m}$. Dacă s-a produs despicarea vreunei clase la acest pas se merge la P2. În caz contrar se va continua.

11. Se calculează toate distanţele între centri claselor

$$d_{ij} = d\left(\underline{m}^i, \underline{m}^j\right) = \left\| \underline{m}^i - \underline{m}^j \right\| \;, \;\; i = \overline{1,k-1} \;, \;\; j = \overline{i+1,k}$$

12. Se compară distanţele d_{ij} cu parametrul θ_c. Se aranjează primele L distanţe mai mici decât θ_c în ordine crescătoare.

13. Pornind de la perechea de clase $\underline{A}_i$, $\underline{A}_j$ având distanţa centrilor cea mai mică se modifică centrii după următoarea regulă: dacă atât pentru $\underline{A}_i$ cât şi pentru $\underline{A}_j$ centrul nu a fost modificat, se înlocuiesc centrii m^i , m^j prin centrul

$$\underline{m}^* = \frac{1}{p_i + p_j}\left(p_i \underline{m}^i + p_j \underline{m}^j\right).$$

Se şterg m^i, m^j şi se pune $k := k-1$. Dacă fie A_i fie A_j au centrul modificat atunci perechea (i, j) nu e luată în considerare şi se trece la următoarea pereche.

14. Dacă aceasta este ultima iteraţie sau dacă poziţia şi numărul centrilor coincid cu cele de la iteraţia precedentă STOP. In caz contrar se merge la 2 (sau 1 dacă se modifică datele iniţiale).

4. Algoritmi genetici şi evolutivi

Algoritmii genetici şi evolutivi reprezintă mai multe clase de metode aleatoare de căutare. Fiecare element implicat în procesul de căutare îl putem descrie printr-o variabilă.

Structura oricărui algoritm de calcul evolutiv [1][2] este, în general:

```
1.  procedura algoritm_evolutiv () este
2.  t <-- 0
3.  creare P(t)
4.  evaluare P(t)
5.  cât timp not condiţia de stop execută
6.      t <-- t + 1
7.      selectare P(t) din P(t-1)
8.      modificare P(t)
9.      evaluare P(t)
10. @
11. sfârsit
```

Conform modelului enunţat mai sus putem sa descriem câteva caracteristici ale algoritmilor genetici [5]. Aceste caracteristici se referă mai ales la particularităţile algoritmilor genetici în raport cu alte metode de căutare şi optimizare.

1. Algoritmii genetici sunt o clasă de algoritmi probabilistici care combină elemente de căutare dirijată şi căutare aleatorie. Aceşti algoritmi realizează un echilibru optim între explorarea spaţiului stărilor şi cele mai bune soluţii identificate.

2. Algoritmii genetici sunt robuşti în comparaţie cu alte metode existente de căutare dirijată sau ca şi algoritmii clasici de optimizare.

3. Metodele de căutare bazate pe algoritmii genetici sunt caracterizate de faptul că ele menţin o mulţime de soluţii potenţiale. Ca şi comparaţie metodele clasice de căutare acţionează la un moment dat doar asupra unui punct din spaţiul de căutare.

4. Algoritmii genetici spre deosebire de cei clasici acţionează cu o codificare asupra elementelor din spaţiul stărilor şi nu acţionează direct asupra spaţiului.

5. Algoritmii genetici utilizează funcţii de performanţă care sunt obţinute prin transformări simple ale funcţiei obiectiv.

6. Algoritmii genetici sunt simplu de utilizat

7. Algoritmii genetici au o mare probabilitate de identificare a soluţiei optime.

Cap 2. Calculul paralel şi algoritmii specifici

2.1 Structuri dedicate de calcul paralel

O structură de calcul paralel reprezintă un sistem de calcul care are blocuri de calcul ce efectuează mai multe operaţii în acelaşi timp. Baza acestor sisteme de calcul paralel este algoritmul paralel.

Deci putem da o definiţie a unui calculator paralel astfel [1]:

- un calculator paralel este o colecţie de procesoare, de regulă de acelaşi tip, interconectate într-o anumită reţea care permite coordonarea activităţilor lor precum şi schimbarea de date.

Scopul procesări paralele este executarea unor calcule mai rapid decât ar fi posibil cu un procesor, prin utilizarea concurentă a mai multor procesoare.

Sunt trei cazuri în care este necesară utilizarea calcului paralel:

1. Pentru a atinge performanţe ridicate în timpul execuţiei unui program

2. Pentru că este o arhitectură disponibilă

3. Pentru că softurile executate au nevoie de un astfel de sistem, scăzând durata execuţiei

În ultima perioada de timp au fost studiate şi construite mai multe tipuri de arhitecturi de calcul paralel, mergând de la structuri cu maşini interconectate cu procesoare puternice până la maşini cu mii de procesoare lente.

Diferenţele între arhitecturile paralele sunt date de câţiva parametri:

1. Numărul de procesoare şi puterea procesorului individual

2. Complexitatea reţelei de conectare şi flexibilitatea sistemului

3. Distribuţia controlului întregului sistem de calcul

4. Mecanismul de control al sistemului

5. Modul de organizare al memoriei

Aşa cum am enumerat mai sus o primă modalitate de clasificare se face în funcţie de numărul de procesoare. Astfel în funcţie de tipul procesoarelor din sistemul paralel, sistemele paralele sunt:

a. Omogene, adică conţin procesoare de acelaşi tip

b. Heterogene, adică conţin procesoare de tipuri diferite

Un alt punct în clasificarea enunţată mai sus este clasificarea din punct de vedere al mecanismului de control. Astfel avem mai multe categorii de sisteme paralele [2]:

1. Grup de procesoare per instrucţiune unică, care la un timp dat numai un set mic de instrucţiuni este într-o anumită fază de de execuţie. Aici avem două categorii principale:

a. Procesoare matriceale care efectuează calculul paralel sincronizat, adică instrucţiuni individuale operează asupra unui număr mare de date.

b. Calculatoare care efectuează mai multe instrucţiuni concomitent

2. Grup de procesoare per instrucţiuni multiple

O altă modalitate de clasificare este din punct de vedere al organizării memoriei

a. Multiprocesoare cu memorie comună

b. Multiprocesoare cu memorie distribuită

Clasificarea după modul de comunicare a datelor între procesoare dintr-o reţea

a. Sistem multiprocesor strâns cuplat, în care procesoarele lucrează împreună pentru rezolvarea unei probleme

b Sistem multiprocesor slab cuplat, este acel sistem în care un număr de procesoare independente şi nu neapărat identice comunică între ele printr-o reţea de comunicare.

În prezent se utilizează un alt mod de clasificare numit clasificarea Flynn. Conform acestei clasificări există patru tipuri de sisteme [2]:

1. SISD : sistem cu un singur set de instrucţiuni şi un singur set de date

2. SIMD : sistem cu un singur set de instrucţiuni şi mai multe seturi de date

3. MISD : sistem cu mai multe seturi de instrucţiuni şi un singur set de date

4. MIMD: sistem cu mai multe seturi de instrucţiuni şi mai multe seturi de date

Sistemele de calcul paralele utilizate în mod curent fac parte din categoriile SIMD şi MIMD.

Astfel în primul caz procesarea paralelă are loc în paşi sincronizaţi, în timp ce în al doilea caz are loc în paşi independenţi.

Procesarea este de tip concurent şi este de două tipuri:

1. Pipelining

2. Paralelism

1. Procesarea de tip pipelining

Un procesor de tip pipeline conţine un număr de procesoare aranjate astfel încât datele de ieşire ale unui procesor să constituie datele de intrare ale altui procesor.

2. Procesarea în paralel

Într-un calculator paralel procesoarele sunt aranjate într-o structură care să permită operarea simultană a mai multor secvenţe de date. În acest tip de procesare operaţiile se realizează prin executarea în secvenţă a mai multor sarcini distincte.

Sisteme de tip SISD

Acest tip de sistem reprezintă de fapt o maşină serială clasică. Prin definiţie acest tip de maşină execută o singură instrucţiune la un moment dat.

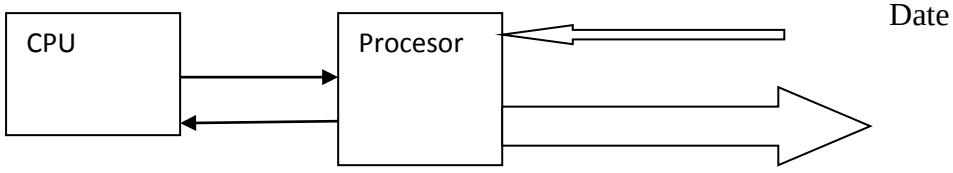


Figura 7 Sistem SISD

O schema detaliata de funcţionare a unui sistem SISD este dată în figura 8:

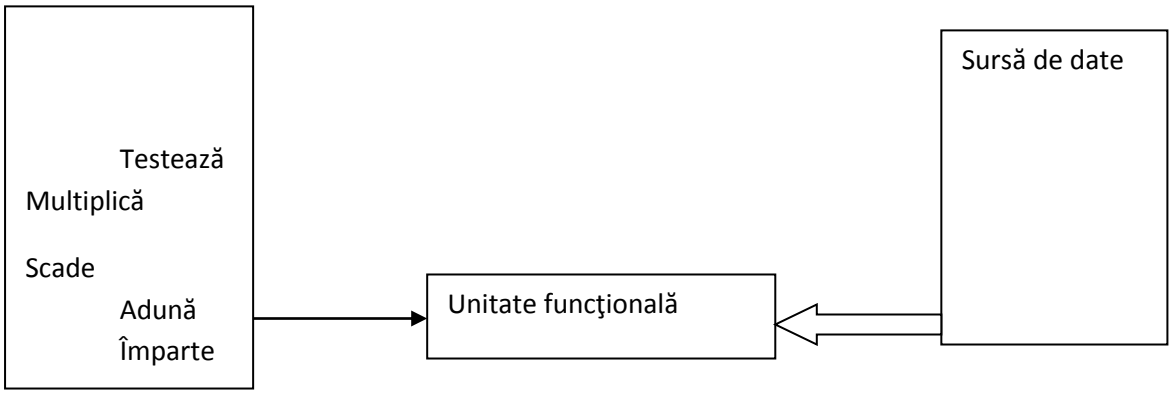


Figura 8 Schema bloc maşină SISD

2. Sisteme de tip SIMD

Un sistem SIMD este de regulă compus dintr-un MCU şi un număr de procesoare identice. În aceste sisteme MCU transmite aceeaşi interacţiune la fiecare procesor în parte. Instrucţiunile sunt executate în acelaşi timp, adică sunt sincrone. Specific acestui tip de sistem este faptul că fiecare procesor are o zonă de memorie proprie. Anumite sisteme permit acccesul la o memorie de tip global, fiecare procesor operând asupra unui cuvânt de 32 sau 64 de biţi sau asupra unui operand de un singur bit pentru fiecare ciclu de memorie.

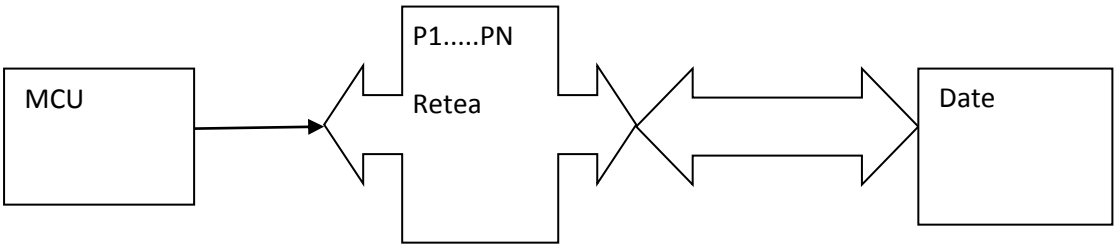


Figura 9 Schemă bloc sistem SIMD

O schemă de funcţionare mai detaliată în figura 10:

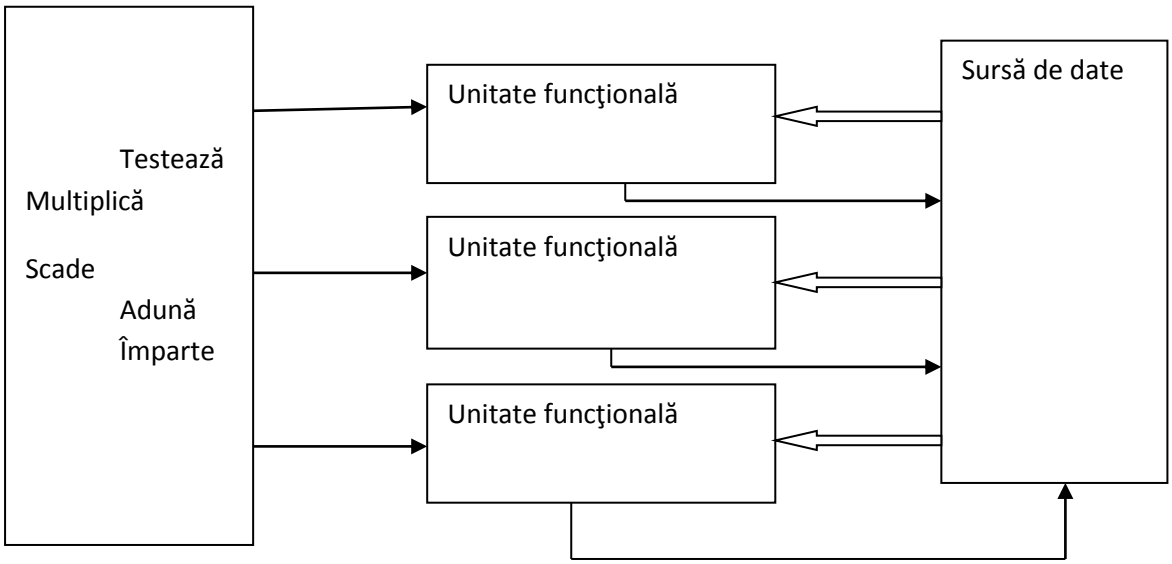


Figura 10 Schemă bloc detaliată SIMD

Aceste tipuri de sisteme sunt utilizate de obicei la rezolvarea unor probleme care pot fi descompuse în subprobleme care necesită un efort de calcul identic, cum ar fi:

- procesarea de imagini

- dinamica fluidelor

- automate de tip celular

De asemenea avem două tipuri distincte de sisteme SIMD:

1. SIMD organizate pe cuvânt, în care operaţiile aritmetice sunt similare cu cele efectuate de maşinile seriale

2. SIMD organizate pe bit, în care algoritmii depind de lungimea în biţi a datelor . La aceste maşini sunt necesare mai multe instrucţiuni pentru efectuarea unei operaţiuni aritmetice simple. În aceste sisteme timpul aritmetic este mai lung decât cel din sistemele organizate pe cuvânt. Creşterea vitezei nu se obţine prin creşterea vitezei fiecărui procesor în parte, ci prin utilizarea a cât mai multor procesoare simultan. Exemplele clasice de SIMD sunt: procesoarele vectoriale, procesoarele matriceale şi câteva tipuri de matrici sistolice.

3. Sisteme de tip MISD

Aceste sisteme sunt de fapt procesoare de tip pipeline care efectuează operaţii asupra unui set mic de date. O schema de baza este dată în figura 11:

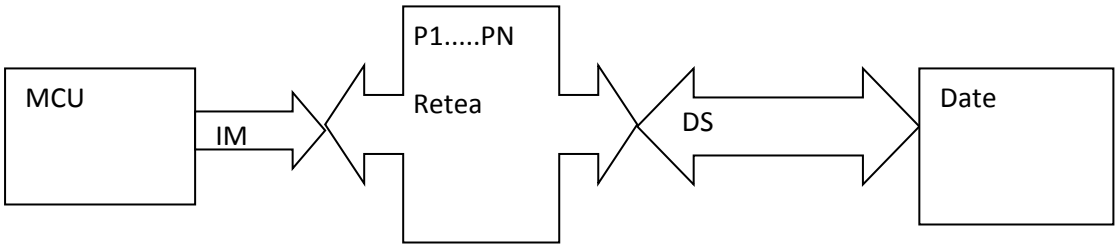


Figura 11 Schemă bloc MISD

4. Sisteme de tip MIMD

Specific acestui tip de sistem este faptul că fiecare procesor poate executa diferite operaţii pe date distincte de cele ale altor procesoare. Deci putem afirma că fiecare procesor funcţionează independent de celelalte utilizând propriul contor de program şi de asemeni setul propriu de instrucţiuni, figura 12:

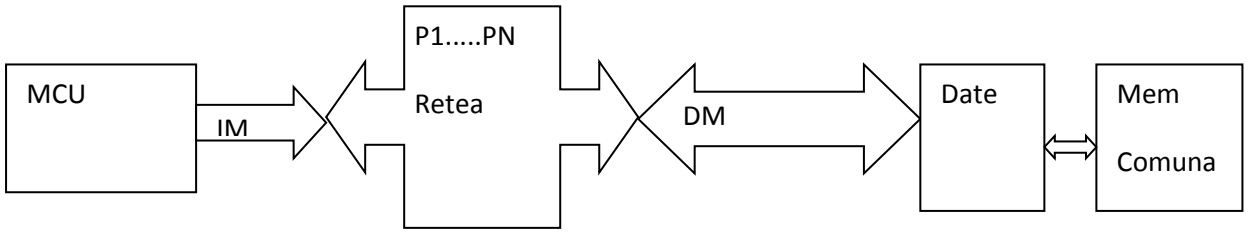


Figura 12 Schemă bloc MIMD

Instrucţiunile specifice unor procesoare distincte sunt independente unele de altele, execuţia unei instrucţiuni neinfluenţând execuţia alteia, deci procesoarele pot opera în orice moment.

Actualele sisteme de tip MIMD au un număr de procesoare mai mic decât sistemele de tip SIMD.

Voi face în continuare câteva consideraţii referitoare la sistemele de tip SIMD şi MIMD. Una dintre consideraţii este legată de configuraţia reţelei. Astfel pentru sistemele SIMD are sincronizarea între elementele din reţea, în cazul nostru procesoarele, se face pe un tact comun. În cazul sistemelor MIMD sincronizarea între elementele de reţea este asincronă, adică fiecare procesor are un tact propriu.

O altă deosebire este dată de modul de execuţie a unui program pe sistemele SIMD şi MIMD.

Astfel pentru un sistem SIMD execuţia uni program depinde foarte mult de setul de date pe care se operează. Astfel în cazul unor instrucţiuni condiţionale aplicate asupra setului de date, execuţia pe două sau mai multe procesoare se face diferit. Rezolvarea acestei probleme în cazul sistemului SIMD se face prin intermediul unui registru permisiune sau enable register

care permite execuţia operaţiilor de scriere. Selecţia se face prin setarea enable register pe modul TRUE. În continuare voi exemplifica un model clasic de algoritm pentru SIMD [2]:

instrucţiune 1

if not *conditie* then

enable= false

instructiune 2

enable=not *enable*

instructiune 3

instructiune 4

enable=True

instructiune 5

instructiune 6

În cazul unui sistem MIMD acesta nu va avea probleme legate de interpretarea expresiilor condiţionale, deoarece fiecare procesor va decide codul propriei instrucţiuni ce va fi executată. În schimb pot apărea şi în cazul acestor sisteme probleme datorită timpilor diferiţi de execuţie.În continuare ca şi în cazul SIMD voi exemplifica cu o structură de algoritm:

instructiune 1

if *conditie* then

instructiune 2

else

{instructiune 3

instructiune 4}

instructiune 5

instructiune 6

O altă consideraţie se poate face pe baza partiţionabilităţii sistemelor. Astfel putem defini un sistem partiţionabil ca fiind acel sistem care permite o partiţionare şi o repartizare dinamică pe maşini multipleşi care permite următoarele:

1. Abilitatea de partiţionare a reţelei în subreţele, fiecare funcţionând ca o reţea completă

2. Independenţa submaşinilor, adică maşinile nu pot interfera fară instrucţiuni adecvate.

Ca şi avantaje de utilizare a unei astfel de sistem putem enumera:

1. Dacă un procesor are probleme, sunt afectate numai acele submaşini care depind de acel procesor

2. Mai multe submaşini pot executa acelaşi program pe acelaşi set de date şi pot compara rezultatele

3. Permite accesul simultan în sistem a mai multor utilizatori care execută diferite programe paralele.

O altă structură în care se pot regăsi sistemele de tip MIMD este structura numită transputer. Astfel putem clasifica aceste sisteme astfel:

1. Sisteme care utilizează elemente de tip switch. Acestea presupun un număr mic de procesoare, conectate printr-o magistrală de date (bus) sau printr-o reţea de switch-uri. Acestea asigură comunicarea între procesoare, funcţionând ca un sistem cu memorie distribuită, comutatoarele memorie procesoare asigură transferul optimal de date între diverse blocuri de memorie şi procesoare. În cazul acestor sisteme procesoarele sunt autonome şi pot opera ca şi un calculator independent. Cu un astfel de sistem se pot rezolva algoritmi paraleli care sunt divizaţi în algoritmi mai mici.

2. Sisteme de tip reţea. Sunt acele sisteme care au un număr mare de procesoare identice care sunt conectate între ele printr-o reţea. Numărul acestora poate fi de la 128 şi până la 65536 procesoare conectate în structuri cunoscute sub denumirea de hipercub. Alegerea structurii de interconectare se face în funcţie de performanţa unui anumit algoritm.

Transputerul este un element de procesare care este construit dintr-un cip microprocesor cu memoria adiţională, unitate aritmetică şi facilitaţi de reţea. În acest sistem procesoarele nu sunt autonome şi sunt dirijate de către un computer extern denumit generic host. Un transputer presupune un număr de patru legături fizice. O schemă generala este dată în figura 13:

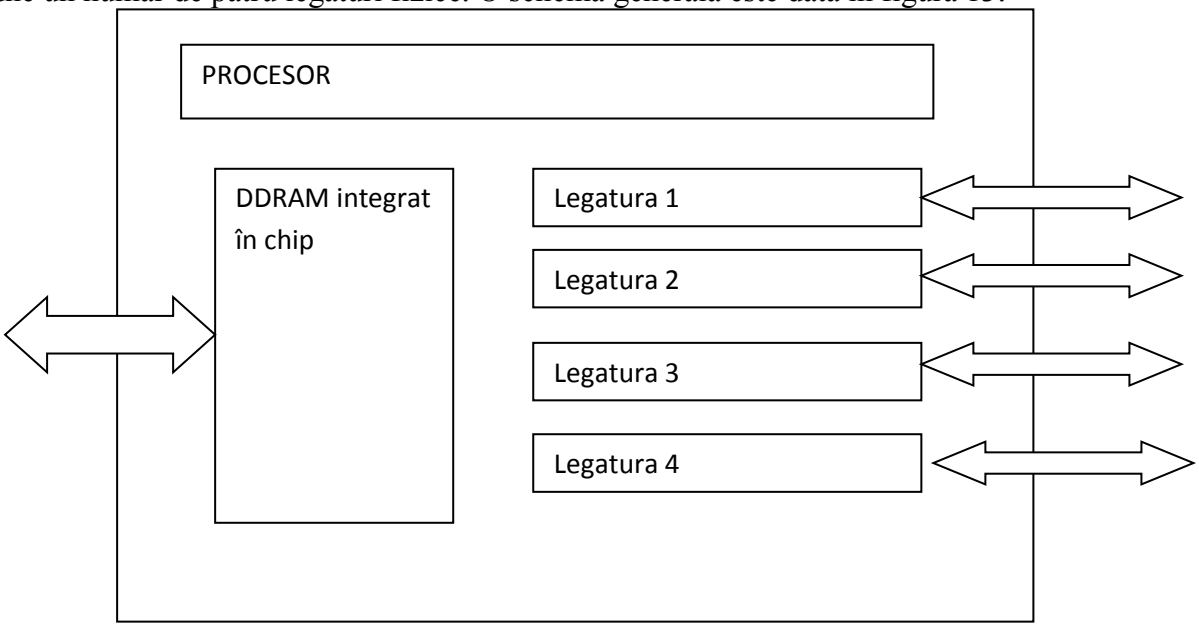


Figura 13 Schemă bloc Transputer

Fiecare legătură este un canal bidirecţional de comunicaţie. Transputerele pot fi interconectate în orice configuraţie dorită.

2.2 Algoritmi specifici calculului paralel

Pentru implementarea unei probleme de calcul paralel primul pas este acela de a descompune procesul astfel încât procesoarele să lucreze concurent la rezolvarea problemei. Există trei metode de generare a algoritmilor paraleli [2]:

1. Metoda unităţilor independente. Prin această metodă fiecare procesor execută acelaşi program independent de celelalte procesoare.

2. Metoda paralelismului geometric. Prin această metodă fiecare procesor execută acelaşi program asupra unor date corespunzătoare unei microstructuri a sistemului.

3. Metoda paralelismului în algoritm. Prin această metodă fiecare procesor este responsabil de câte o parte dintr-un algoritm şi de toate datele care trec prin fiecare procesor.

Putem discuta în cazul algoritmilor specifici de algoritmi paraleli fundamentali. Acestia sunt [2]:

a. Algoritmul " Divide et Impera"[3]. Implementarea acestui algoritm presupune împărţirea problemei în subprobleme mici care pot fi tratate independent. Strategia de implementare a algoritmului constă în trei paşi principali:

1. Partiţionarea problemei în subprobleme de dimensiuni egale

2. Rezolvarea recursivă a subproblemelor şi rezolvarea concurentă a întregului set de subprobleme

3. Combinarea soluţiilor subproblemelor într-o soluţie pentru problema generală

b. Algoritmul bazat pe multiplicarea a două matrici

c. Algoritmi pentru sisteme organizate pe biţi

d. Algoritmi bazaţi pe grafuri orientate

e. Algoritmi de sortare

f. Algoritmi de clasare

Voi exemplifica în continuare cum se pot construi algoritmi specifici calculului paralel. Astfel pentru a modela algoritmii paraleli numerici există o serie de căi:

a. Analiza algoritmului serial şi convertirea sa printr-o procedură care va opera cu vectori şi matrici astfel încât majoritatea datelor să fie prelucrate simultan.

b. Algoritmi iterativi echivalenţi care prin anumite metode sunt transformaţi sistematic pe un calculator paralel

c. Calculul poate fi divizat în subunităţi şi distribuit între procesoare.

Cap 3 . Sisteme de calcul paralel utilizate în implementarea algoritmilor paraleli

3.1 Clasificarea sistemelor paralele şi distribuite

În acest capitol voi face o trecere în revistă a câtorva sisteme de calcul paralel care sunt utilizate la implementarea unor algoritmi paraleli.

Voi începe cu scurtă clasificare a sistemelor de calcul paralel. Astfel avem în funcţie de numărul de unităţi centrale conectate la memorie:

1. Clasa SCPU sau Single Central Processing Unit. Acesta aşa cum îl arată şi numele are o arhitectură bazată pe o singură unitate centrală legată la memorie. Mai este cunoscută şi sub denumirea de arhitectură von Neumann

Figura 14 Schemă bloc Von Neumann

2. Clasa MCPU sau Multiple Central Processing Unit. Acesta are mai multe unităţi centrale legate la memorie. Structura mai poartă numele de structură non von Neumann.

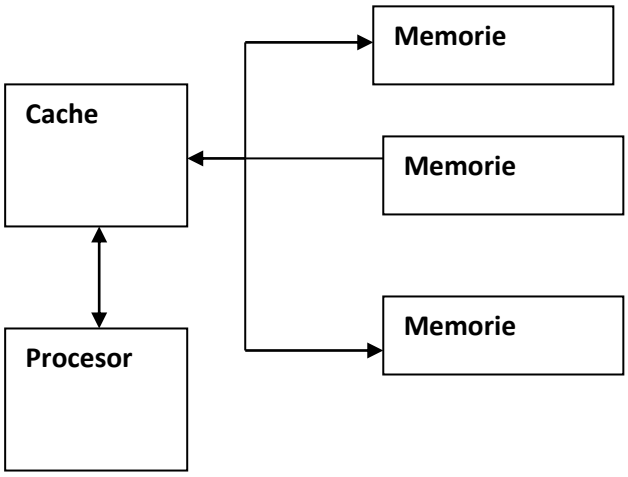


Figura 15 Schemă bloc MCPU

O altă clasificare se poate face după mecanismul de control. Astfel putem menţiona două categorii principale:

1. Clasa Global Control Unit sau prescurtat GCU. Este o arhitectură care are la bază un singur CPU care execută controlul asupra întregului sistem

2. Clasa Local Control Unit. Este o arhitectură care are mai multe procesoare locale care controlează execuţia proceselor locale.

Se poate face de asemenea o clasificare şi după numărul de instrucţiuni prelucrabile:

1. Clasa Single Instruction Stream, în care sistemul va executa doar un singur set de instrucţiuni.

2. Clasa Multiple Instruction Stream, în care sistemul este capabil să execute mai multe seturi de instrucţiuni.

În continuare voi încerca să descriu câteva structuri de calculatoare, numite calculatoare scalare şi modul de implementare a paralelismului în aceste structuri.

Aşa cum am menţionat în capitolul anterior calculatoarele paralele sunt SISD, MIMD,SIMD şi MISD.

Un calculator de tip SISD mai este denumit şi calculator scalar deoarece au o structură pseudoparalelă. Aceasta înseamnă că deşi execută numai programe seriale, aceste sisteme

înglobează foarte multe concepte de paralelism virtual, în special în procesoare. Viteza de calcul a unui astfel de sistem este influenţată de viteza de execuţie a instrucţiunilor.

În figura 16 am exemplificat acest concept [1][2]:

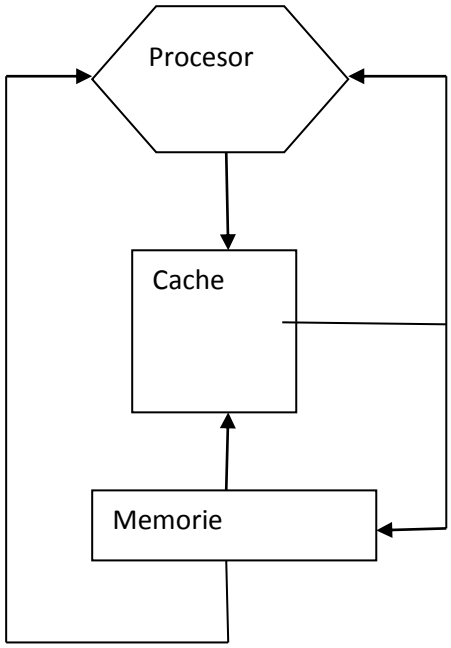


Figura 16 Schemă bloc SISD

Aceste structuri au în schimb o problemă care poartă denumirea de limitare von Neumann. Pentru a se elimina această limitare există câteva soluţii menţionate de literatura de specialitate:

a. Utilizarea de memorie întreţesută. Această metodă permite accesarea mai multor date din memorie simultan de către procesor

b. Utilizarea memoriei cache. Această metodă datele care urmează a fi prelucrate se stochează într-o zonă de memorie tampon, numită cache. Această zonă de memorie fiind mai apropiată de procesor permite o creştere a vitezei de procesare.

c. Utilizarea concomitentă a două zone de memorie cahe.

d. Pipelining-ul este acea metodă prin care creşterea vitezei de execuţie a instrucţiunilor se face prin procesarea paralelă în timp a instrucţiunilor.

e. Multiprogramarea este un concept de natură software care constă tot în execuţia aparent simultană a mai multor programe în acelaşi interval de timp, dar urmarind o planificare .

f. Randomizarea care este un paralelism simulat software sau hardware prin multiprelucrare distribuită.

Putem discuta şi de structuri multiscalare. Acestea sunt de regulă structurile SIMD sau MIMD. Astfel spre deosebire de structurile de tip SISD, care aşa cum am arătat anterior procesează un singur set de instrucţiuni pe un singur set de date prin procesări paralele cu programe seriale, în arhitecturile de tip multiscalar SIMD, avem un paralelism avansat care permite execuţia unui set de instrucţiuni pe mai multe seturi de date.

În acest tip de arhitectură avem mai multe procesoare care sunt puse să execute simultan pe baza unui program paralel-sincron o aceeaşi instrucţiune, fiecare pe date distincte. Aceste procesoare vor comunica între ele prin intermediul memoriei partajate. Modul de lucru este următorul: comanda întregului sistem este asigurat de un bloc de comanda şi control care analizează datele, dacă acestea sunt simple le execută iar dacă sunt multiple le distribuie la celelalte procesoare efortul de calcul fiind similar pe fiecare procesor.

O altă modalitate de clasificare este clasificarea Flynn. Această clasificare constă din combinaţii ale claselor SI şi MI cu clasele SD şi MD.

Aici putem include pe lângă calculatoarele scalare şi multiscalare precizate anterior calculatoarele virtuale şi calculatoarele compact paralele.

Calculatoarele virtuale sunt calculatoare care pot prelucra seturi multiple de date, de exemplu un calculator MISMD sau reţelele de calculatoare.

Calculatoarele compact paralele sunt calculatoare de tip MIMD multiprocesor .

Mai nou pentru a se putea mări puterea de calcul şi pentru a creşte eficienţa calculatorului serial s-au adoptat trei soluţii:

1. Creşterea vitezei de operare a componentelor

2. Paralelizarea procesului de calcul

3. Specializarea calculatoarelor pentru rezolvarea unei anumite clase de probleme.

Este cunoscut faptul că calculatoarele paralele au o viteză şi o capacitate superioară calculatorului serial.

Se ştie de asemenea că calculatoarele paralele au o viteză şi o capacitate de stocare care este clar superioară calculatorului serial.

Pentru a putea rezolva toate problemele enunţate mai sus s-a trecut la un concept nou care permite aplicarea unor soluţii pentru a rula mai rapid aplicaţiile şi care permit distribuirea sarcinilor de lucru pe mai multe calculatoare aflate într-o reţea. Aceste sisteme sunt cunoscute în literatura de specialitate sub denumirea de cluster şi grid.

Astfel putem defini cele două concepte, respectiv cluster computing şi grid computing:

- ***Cluster computing*** reprezintă o colecţie de staţii de lucru, omogene sau neomogene, cuplate într-o reţea bine localizată, care are un control centralizat, accesibil de la diverse terminale numite staţii de lucru şi care sunt administrate de un software centralizat.

- ***Grid computing*** reprezintă o colecţie de resurse informatice eterogene care nu sunt localizate într-un spaţiu anumit , care nu are un anume control centralizat, fără a avea o imagine de sistem unic şi care poate fi accesat printr-o anumită platformă software.

-***Mediogrid***

Arhitectura sistemului MEDIOGRID [7][8][9]prezentată în figura 17 are în compunere următoarele elemente:

- Sistemul de achiziţie a imaginilor. Acesta are rolul de a prelua şi apoi a stoca imaginile

- Sistemul de prelucrare a datelor primare. Acesta are rolul de a compune imaginile și corespondența de date a unor benzi spectrale și apoi le transmite la serverul de aplicații.

- Serverul de aplicaţii. Acesta comunică cu sistemul primar de prelucrare a datelor, împarte imaginile, reasamblează și transmite proceselor rezultatele finale

- Serverul de tip GRID [8][9]. Acesta administrează infrastructura rețelei, primește date de la serverul de aplicație și le trimite în rețea pentru a fi calculat, primește rezultatele și le întoarce la serverul de aplicații.

- Grid-ul client are rolul de a efectua procesele de bază privind specificarea datelor analizate.

Deci aşa cum se poate observa o structura de tip MEDIOGRID poate rezolva atât cerinţele de hardware specifice unor sisteme de analiză rapidă a datelor cât şi software prin aplicaţiile pe care le poate rula prin sistemele de tip client.

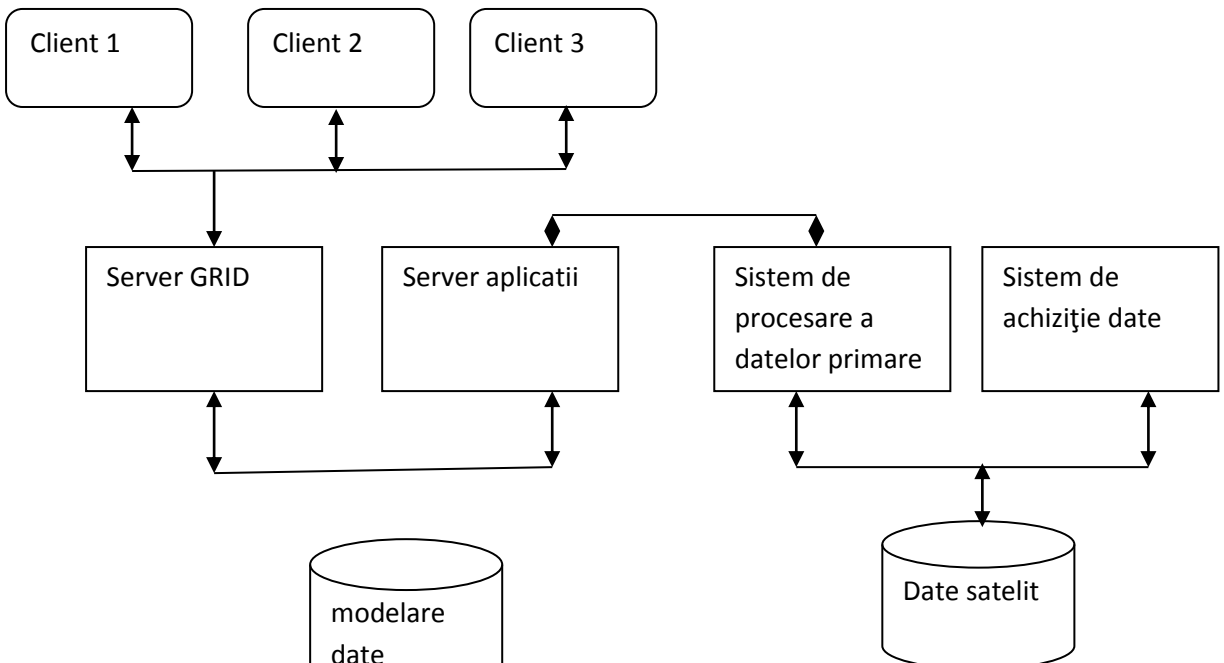


Figura 17 Schema bloc MEDIOGRID

Cap. 4 Aplicaţii practice actuale ale algoritmilor paraleli utilizaţi în recunoaşterea formelor

În acest capitol voi încerca să prezint câteva aplicaţii ale unor algoritmi paraleli uzuali. Datorită faptului că interesul de cercetare în acest moment este pentru structurile de tip JAVA voi trata în special algoritmii paraleli şi distribuiţi folosiţi în aplicaţiile de tip JAVA. Aceşti algoritmi sunt asemănători cu cei clasici dar au particularităţile structurale ale limbajului JAVA.

Astfel, se cunoaşte faptul că :

- Limbajul JAVA este *orientat obiect*. Cu el se pot crea clase de obiecte si instante ale acestora, se pot incapsula informatiile, se pot mosteni variabilele si metodele de la o clasa la alta. Singura caracteristica care lipseste este mostenirea multipla, dar pentru a suplini aceasta lipsa, Java ofera o facilitate mai simpla, numita interfata, care permite definirea unui anumit comportament pentru o clasa de obiecte, altul decat cel definit de clasa de baza. In Java orice element este un obiect, in afara de datele primare. Din Java lipsesc functiile si variabilele globale.

- Limbajul JAVA este ***distribuit***, avand implementate biblioteci pentru lucrul in retea.

- Limbajul JAVA este un limbaj cu *securitate ridicata*

Am subliniat faptul că limbajul JAVA este distribuit tocmai pentru a pune în evidenţă ce am precizat la început. Un exemplu de algorim si foarte cunoscut este algoritmul K-MEANS care poate fi implementat atât în C cât şi în JAVA, respectând binenînţeles caracteristicile de programare.

Un prim exemplu de utilizare a algoritmilor paraleli în aplicaţii ar putea fi modelul PVM-MPI care mai este cunoscut şi ca programare cu JCLUSTER [1]. Astfel ştim că realizarea alicaţiilor paralel distribuite într-un cluster sau într-o reţea de staţii de lucru are la bază, de regulă, algoritmul MPI. Prin utilizarea limbajului de programare JAVA a fost dezvoltată o aplicaţie numită Jcluster care implementează un algoritm derivat din MPI cunoscut ca algoritmul MPI-PVM. Aplicaţiile acestui algoritm se bazează pe două modele:

- modelul master-slave

- modelul nodal

Un alt produs derivat este BSP (Bulk Synchronous Parallel) [1]. Aceasta este în esenţă o interfaţă de programare dar are la bază algoritmul de tip MPI.

Un alt model cunoscut este modelul TSpaces sau modelul Linda. Acesta este o implementare a unei memorii partajabile şi a fost dezvoltat de către firma IBM. Acest model de calcul, cunoscut şi sub numele de Linda, a fost dezvoltat pentru implementare în cadrul standardului JavaSpaces. În acest model, un proces poate depune date, care apoi sunt regăsite în alte procese, realizându-se astfel comunicaţia dintre procese.

Există de asemenea şi o altă implementare a algoritmilor paraleli pentru JAVA numită Java Communicating Sequential Processes. Modul de construcţie a unui program are la bază modul de programare în limbajul OCCAM, care în esenţă are la bază un model bazat pe CSP şi care este dedicat unui procesor specializat numit transputer.

De asemenea putem menţiona JAVA OpenMP sau prescurtat JOMP [1]. Acesta este o transpunere în JAVA a unei metode de paralelizare a programelor dezvoltată de către Edinburgh Parallel Computing Center şi are la bază modele din Fortran şi C++. Această

metodă constă în introducerea într-un program secvenţial a unor directive care transformă programul într-un program paralel.

Bibliografie:

1.Ernest Scheiber " Programare concurenta si paralel distribuita in JAVA" Editura Albastra Cluj Napoca 2007

2.I.Dzitac " Sisteme distribuite. Modele Informatice" Editura Universitatii Agora, Oradea 2006

3.Eleonor Ciurea, Cristina Luca " Algoritmi si programare in Java. Teorie si aplicati" Editura Albastra, Cluj Napoca 2005

4.Adrian Deaconu " Programarea in limbajele C/C++ si aplicatii" Editura Albastra, Cluj Napoca 2007

5.I.Dzitac " Sisteme de recunoastere a formelor. Note de curs" Universitatea Agora Oradea

6. . D.Gorgan, O.Muresan " Satellite Image processing by MedioGRD Platform Kernel" Proceedings CSCS 16

7. D.Tudor, V.Cretu " A view on fault tolerant techniques applied for Mediogrid" Proceedings of the International Conference on knowledge engineering, Cluj Napoca June 6-8 2007

8. Bacu V., Muresan O. "Modis Image based computation of vegetation indices in Mediogrid architecture" Proceedings of the SYNASC 2006, Timisoara

9. V.Bacu, D.Gorgan " Graph Based Evaluation of satellite imagery procesing over grid" Ucluj Mediogrid