METODE DE CLASIFICAREA SINTACTICĂ A FORMELOR PLANE DESCRISE PRIN STRUCTURI DE TIP ARBORE

Ovidiu Grigore
Octavian Grigore

Rezumat: O dată cu dezvoltarea domeniului recunoaşterii sintactice a formelor, precum şi al domeniului analizei şi al descrierii imaginilor a început să se folosească din ce în ce mai mult reprezentarea formelor prin structuri de tip arbore. Folosirea acestor structuri de date conduce inevitabil la probleme de comparare şi de potrivire a arborilor, care se rezolvă prin intermediul definirii unor distanţe între arbori. În lucrare prezentăm o metodă directă (Lu) de calculare a acestei distanţe şi trei metode indirecte, care transformă, mai întâi, arborii în şiruri de primitive, după care se calculează distanţa dintre şirurile respective. Metodele sunt comparate pe baza rezultatelor obţinute într-o aplicaţie de recunoaştere a caracterelor scrise de mână.

Cuvinte cheie:

1. Algoritm direct de calculare a distanţei între arbori

Acest algoritm este denumit direct deoarece calculează distanţa între arbori, plecând de la structura iniţială, de tip arbore, fără a apela la transformări intermediare.

În esenţă, acest algoritm este o generalizare a algoritmului Wagner - Fisher de calculare a distanţei între şiruri. Astfel, în mod similar algoritmului Wagner - Fisher pentru şiruri, a calcula distanţa între doi arbori înseamnă a determina costul minim, necesar transformării primului arbore în cel de-al doilea.

Înainte de a prezenta algoritmul lui Lu, de calculare a distanţei între arbori, vom defini câteva noţiuni care sunt necesare pentru înţelegerea acestuia.

Fie $U = (\mathbf{N} , .)$ monoidul generat de mulţimea numerelor naturale împreună cu operatorul " . ". Elementele lui U sunt de forma $x . y$, în care $x \in U$, iar $y \in \mathbf{N}$. Elementul neutru al monoidului este 0, şi deci:

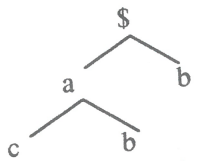
$0 . x = x . 0 = x$

pentru orice $x \in U$.

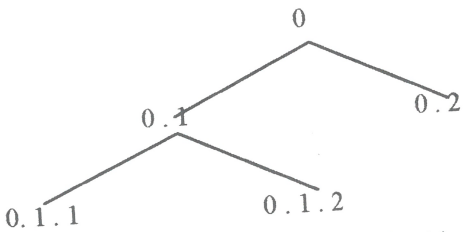
Se defineşte domeniul unui arbore o submulţime $D \subset U$ care satisface condiţiile:

1. dacă $a . x \in D$, unde $a \in U$ şi $x \in U$, atunci $a \in D$;
2. dacă $a . j \in D$, atunci pentru orice $i < j$, $i, j \in \mathbf{N}^+$, avem: $a . i \in D$;
3. elementul neutru al monoidului, notat 0, face parte din mulţimea D.

A ataşa unui arbore γ un arbore ordonat D_γ definit pe domeniul D înseamnă a defini o corespondenţă (funcţie): α: $D \rightarrow \Sigma$, unde Σ este mulţimea etichetelor arborelui iniţial. Pentru a înţelege bine trebuie să consideram următorul exemplu: dacă arborele iniţial γ este:



atunci acestuia i se pune în corespondenţă arborele ordonat D_γ:



cu observaţia că prefixele " 0 . " de obicei nu se mai scriu în faţă etichetelor nodurilor arborelui ordonat, deoarece 0 este elementul neutru al operaţiei " . " şi deci:

$0.1 = 1$

$0.2 = 2$

0.1.1 = 1.1,

0.1.2 = 1.2

Legătura între γ şi D_γ este dată de funcţia

α: D $\rightarrow \Sigma$, definită prin:

α (0) = $

α (1) = a

α (2) = b

α (1 . 1) = c

α (1 . 2) = b

unde: D = { 0, 1, 2, 1 . 1, 1 . 2 },

Σ = { $, a, b, c }

Se notează cu N(γ) numărul de noduri ale unui arbore γ. În exemplul anterior, numărul de noduri este:

N (γ) = 5

Pentru orice arbore γ se defineşte funcţia *rang* r: $D_\gamma \rightarrow \mathbf{N}$, r (*a*) = n, care asociază oricărui nod *a* al arborelui ordonat ataşat D_γ numărul de descendenţi direcţi n. Dacă *b* este un nod terminal, atunci r (*b*) = 0.

Fie două noduri *a, b* din domeniul D. Se spune că între nodurile *a* şi *b* există relaţia de ordonare $a < b$ dacă există x $\in$U astfel încât *a* . x = b. Dacă x = 0, atunci $a = b$. Într-o ordonare $a < b$, *a* se numeşte *predecesor* al lui *b*, iar *b* este *descendentul* lui *a*. Dacă între *a* şi *b* nu există nici una dintre operaţiile de ordonare:

$a < b$,

$a = b$,

$b < a$,

atunci se spune că nodul *a* este *incomparabil* cu *b*.

Se defineşte operatorul $h_\gamma : D_\gamma \rightarrow \mathbf{N}^+$, care ordonează în ordine crescătoare a sufixului secvenţa de noduri din arborele ordonat D_γ ataşat arborelui iniţial γ. Pentru exemplul anterior, în care D_γ = { 0, 1, 2, 1 . 1, 1 . 2 }, această ordonare este:

h_γ (0 . 1 . 1) = 1

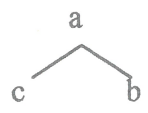
h_γ (0 . 1 . 2) = 2

h_γ (0 . 1) = 3

h_γ (0 . 2) = 4

h_γ (0) = 5

Dacă γ este un arbore ordonat şi $a \in D_\gamma$ este un nod al său, atunci prin γ/a se notează subarborele lui γ care are ca rădăcină pe *a*. În cazul exemplului anterior, subarborele $\gamma/1$ este:



A compara doi arbori α şi β, înseamnă a determina costul minim al modificărilor ce trebuie făcute pentru a transforma primul arbore în cel de-al doilea.

Modificarea (transformarea) etichetei nodului *a* al arborelui α cu cea a nodului *b* al arborelui β se defineşte ca o funcţie: T: $D_\alpha \rightarrow D_\beta$ care realizează următoarele trei tipuri de operaţii:

a) *substituirea (înlocuirea)* etichetei nodului *a* prin cea a lui *b*, notată $a \rightarrow b$, care este cotată cu costul *r*;

b) *inserarea* unui nod *b* în arborele β, echivalentă cu operaţia $\Lambda \rightarrow b$, care are costul *q*;

c) *ştergerea (eliminarea)* nodului *a* din arborele α, adică înlocuirea de tipul $a \rightarrow \Lambda$ a cărei cost este *p*;

în care:

$a \in D\alpha$,

$b \in D\beta$,

Λ este caracterul vid

În cazul substituirii, costul r este nul dacă etichetele celor două noduri a şi b coincid, adică $\alpha(a) = \beta(b)$ şi respectiv $r > 0$, dacă ele nu coincid, $\alpha(a) \neq \beta(b)$.

Distanţa între doi arbori α şi β, notată cu $d(\alpha, \beta)$, este costul minim necesar pentru a transforma arborele α în β folosind o serie de transformări de tipul T, făcute astfel încât să se satisfacă următoarele criterii:

a) dacă $a < b$ în $D\alpha$ atunci $T(a) < T(b)$ în $D\beta$, iar dacă a şi b sunt incomparabile în $D\alpha$ atunci $T(a)$ şi $T(b)$ sunt incomparabile în $D\beta$;

b) dacă $a = b$ în $D\alpha$, atunci $T(a) = T(b)$ în $D\beta$, iar dacă $a \neq b$ în $D\alpha$, atunci $T(a) \neq T(b)$ în $D\beta$;

c) dacă $h_\alpha(a) < h_\alpha(b)$, atunci $h_\beta(T(a)) < h_\beta(T(b))$ cu condiţia ca $T(a)$ sau $T(b)$ să nu fie simboluri nule (vide).

Condiţiile impuse mai sus pentru procesul de modificare a nodurilor unui arbore restrâng numărul de posibilităţi de transformare între care se caută cea de cost minim, însă ele au fost impuse din necesitatea de a păstra pe cât este posibil structura iniţială a arborelui. Astfel, în urma procesului de transformare a unui arbore:

a) trebuie să se păstreze relaţia predecesor-descendent dintre noduri;

b) nu trebuie să apară uniri sau divizări ale nodurilor;

c) se va păstra ordonarea iniţială a sufixului arborelui ordonat.

Vom prezenta, în continuare, algoritmul de calculare a distanţei între doi arbori α şi β. Metoda foloseşte programarea dinamică pentru a calcula elementele unei matrice a distanţelor:

$$\{ D(i,j) \}_{\substack{i=1,n \\ j=1,m}}$$

unde i şi j sunt indicii din ordonarea crescătoare a sufixului nodurilor arborelui β şi respectiv α, iar dacă acestor indici le corespund nodurile a şi b, avem deci $h_\alpha(a) = j$, $h_\beta(b) = i$. Valoarea D(i,j) reprezintă costul minim necesar obţinerii subarborelui α/a din arborele β/b. Numărul de noduri al arborelui α este:

$N(\alpha) = m$,

iar al arborelui β este:

$N(\beta) = n$.

Algoritmul Lu pentru calcularea distanţei între doi arbori [17]:

P1. Iniţializare:
$D(0, 0) = 0$

P2. Pentru $j = 1, m$ calculează:
$a = h_\alpha^{-1}(j)$
$D(0, j) = N(\alpha/a) \times q$

P3. Pentru $i = 1, n$ calculează:
$b = h_\beta^{-1}(i)$
$D(i, 0) = N(\beta/b) \times p$

P4. Pentru $i = 1, n$ execută
$a = h_\alpha^{-1}(j)$
Pentru $j = 1, m$ execută:
$b = h_\beta^{-1}(i)$
$E(0, 0) = 0$
Pentru $l = 1, t$ calculează:
$t = r(a)$
$E(0, l) = E(0, l-1) + N(\alpha/a.l) \times q$

$E(0, t+1) = E(0, t) + q$

Pentru $k = 1, s$ execută:

$s = r(b)$

$E(k, 0) = E(k-1, 0) + N(\beta / b.k) \times p$

$E(s+1, 0) = E(s, 0) + p$

Pentru $k = 1, s$ calculează:

$u = h_\beta(b.k)$

Pentru $l = 1, t$ execută:

$v = h_\alpha(a.l)$

$$E(k,l) = \min \begin{cases} E(k-1,l) + N(\beta/b.k) \times p \\ E(k,l-1) + N(\alpha/a.l) \times q \\ E(k-1,l-1) + D(u,v) \end{cases}$$

Pentru $l = 1, t$ calculează:

$v = h_\alpha(a.l)$

$$E(s+1,l) = \min \begin{cases} E(s+1,l-1) + N(\alpha/a.l) \times q \\ E(s,l) + p \\ E(0,l-1) + D(i,v) \end{cases}$$

Pentru $k = 1, s$ calculează:

$u = h_\beta(b.k)$

$$E(k,t+1) = \min \begin{cases} E(k-1,t+1) + N(\beta/b.k) \times p \\ E(k,t) + q \\ E(k-1,0) + D(u,j) \end{cases}$$

Dacă $\alpha(a) = \beta(b)$ atunci

$r = 0$

altfel

$r = C(a, b)$

$$D(i,j) = \min \begin{cases} E(s,t+1) + p \\ E(s+1,t) + q \\ E(s,t) + r \end{cases}$$

P5. $d(\alpha, \beta) = D(n, m)$

P6. *STOP*

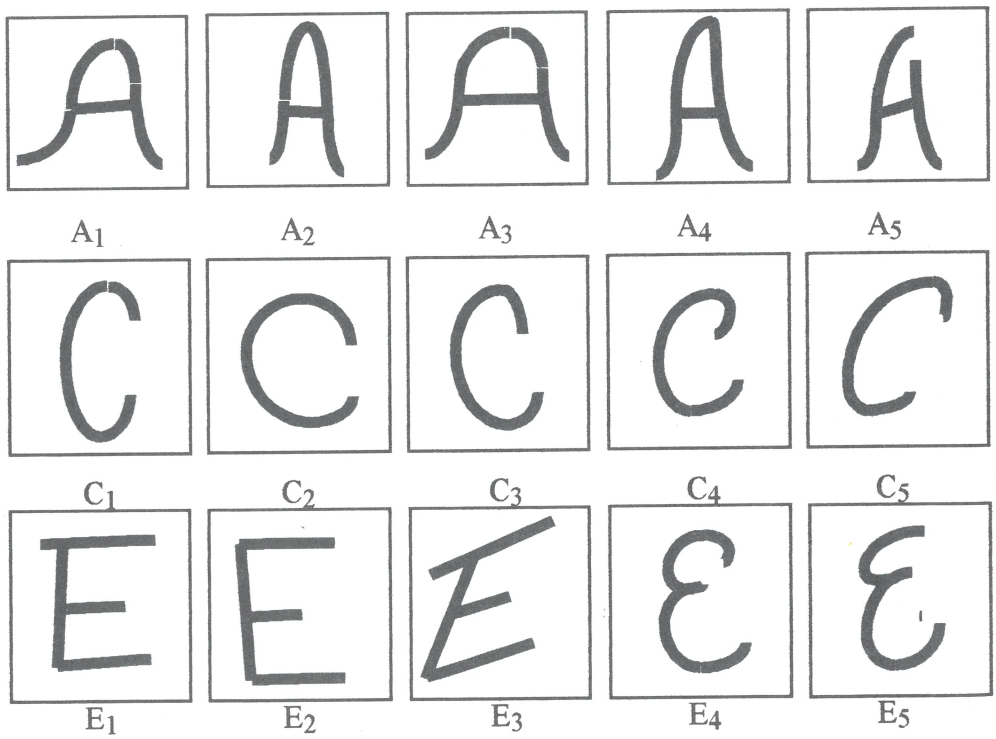


Figura. 1 Lotul de caractere folosit pentru clasificarea nesupervizată pe baza distanţelor între arbori

Metodele nesupervizate ("clustering") sunt, adesea, folosite în recunoaşterea sintactică a formelor. În general, aceste metode folosesc ca măsurare a similarităţii între forme o distanţă în spaţiul de reprezentare al formelor.

În scopul verificării eficienţei algoritmului descris anterior s-a realizat o aplicaţie de clasificare pe bază de distanţe ("clustering") a caracterelor alfa-numerice scrise de mână.

Este folosit un set de date identic cu cel din [17] şi care conţine 15 forme: 5 caractere din clasa "A", 5 din clasa "C" şi 5 din clasa "E". Formele din acest lot sunt prezentate în figura 1.

Caracterele iniţiale sunt supuse, mai întâi, unei etape de preprocesare, care constă în două operaţii succesive: mai întâi, se realizează operaţia de subţiere (thinning) [11], în urma căreia se obţin scheletele formelor de intrare, reprezentate prin linii curbe subţiri, urmând ca apoi acestea să fie aproximate poligonal, pe baza unui anumit criteriu de eroare.

Tabel 1 Codificarea primitivelor folosite pentru descrierea caracterelor

Codul primitivei	Domeniul Unghiular $[\alpha_1, \alpha_2)$	
	α_1	α_2
a	$-7\pi/8$	$-5\pi/8$
b	$-5\pi/8$	$-3\pi/8$
c	$-3\pi/8$	$-\pi/8$
d	$-\pi/8$	$\pi/8$
e	$\pi/8$	$3\pi/8$
f	$3\pi/8$	$5\pi/8$
g	$5\pi/8$	$7\pi/8$
h	$7\pi/8$	$9\pi/8$

Fiecare segment de dreaptă al aproximării poligonale reprezintă o primitivă, iar codificarea lor se face în funcţie de direcţia pe care o au faţă de direcţia orizontală.

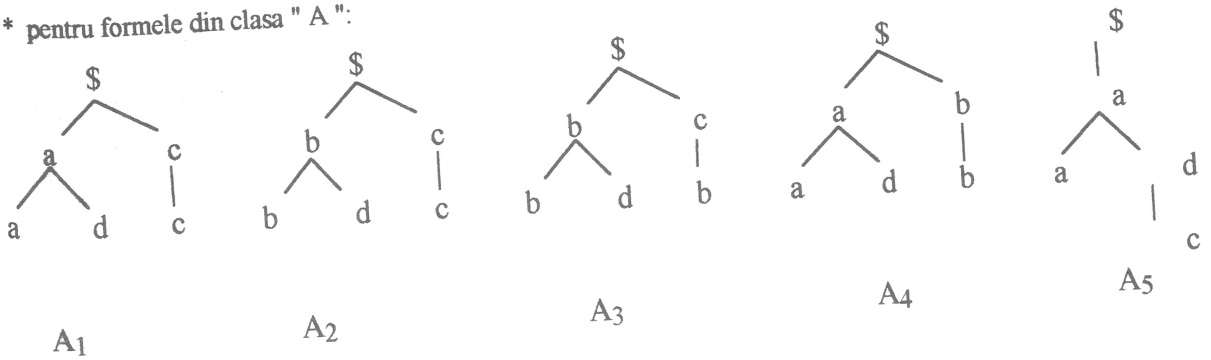
În tabelul 1, este prezentată lista codurilor atribuite primitivelor care se folosesc în descrierea formelor de intrare, împreună cu domeniul unghiular pe care îl reprezintă. Astfel, dacă α este unghiul făcut de un segment de dreaptă ale aproximării poligonale cu axa orizontală, atunci codul primitivei va fi cel corespunzător intervalului pentru care:

$\alpha \in [\alpha_1, \alpha_2)$

Legăturile care există între segmentele de dreaptă, care aproximează caracterele sunt folosite pentru construirea arborelui de primitive, ce constituie structura de date ce va fi prelucrată în continuare.

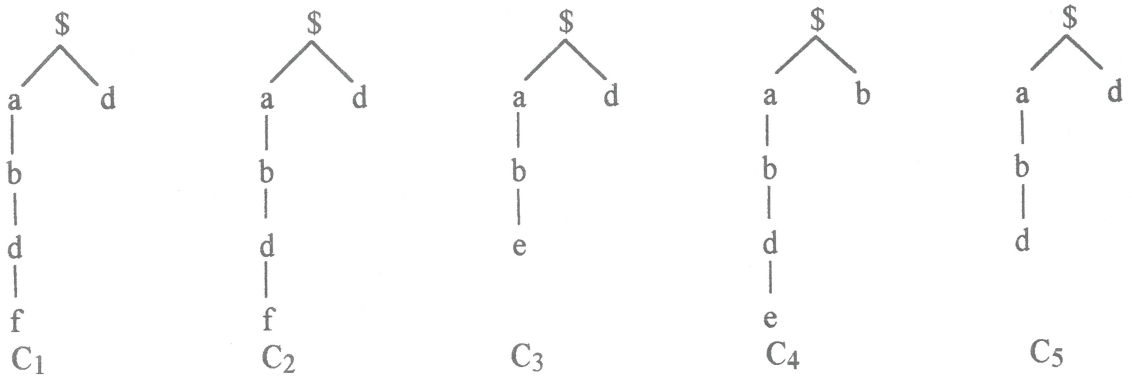
În urma procesului de prelucrare, descris mai sus, celor 15 caractere din lotul utilizat le corespund următoarele reprezentări de tip arbore:

* pentru formele din clasa " A ":

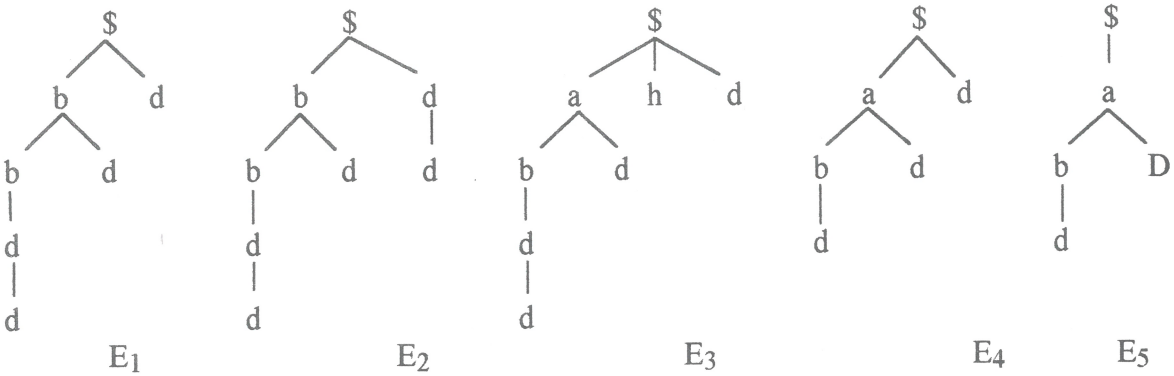


A1 A2 A3 A4 A5

*pentru formele din clasa "C":



*pentru formele din clasa "E":



În tabelul 2 sunt calculate valorile medii ale distanţelor în interiorul fiecărei clase şi între perechi de clase.

Tabel 2 Valorile medii ale distanţelor în interiorul fiecărei clase şi între perechi de clase atunci când s-a aplicat algoritmul Lu

	Clasa "A"	Clasa "C"	Clasa"E"
Clasa "A"	2.48	4.44	4.92
Clasa "C"	4.44	1.44	3.72
Clasa "E"	4.92	3.72	2.32

Observând valorile din tabele se poate afirma că, în general, perechile de eşantioane ale aceleiaşi clase sunt caracterizate de valori mici ale distanţei, iar între formele din clase diferite aceasta are valori mari, însă sunt şi excepţii când distanţa dintre eşantioane din clase diferite se apropie destul de mult de distanţa dintre formele aceleiaşi clase, uneori devenind chiar egală, cum ar fi cazul:

$d(A_5, A_1) = d(A_5, E_5) = 3$

Acest lucru poate fi explicat prin asemănarea fizică între caracterele respective. Astfel, în exemplul de mai sus, A5 este un caracter deformat (distorsionat), care seamănă suficient de bine cu litera E şi pentru un observator uman.

2. Metode indirecte de calculare a distanţei între arbori

Metodele care vor fi prezentate în continuare sunt caracterizate prin faptul că determină distanţa dintre doi arbori prin intermediul transformărilor arborilor în şiruri şi calcularea distanţei între şirurile obţinute astfel.

Vor fi prezentate trei metode rezultate prin combinarea metodelor de transformare a arborilor în şiruri şi a algoritmilor de calculare a distanţei între şirul de primitive, metoda clasică (Wagner-Fisher) şi cea normată.

2.1. Distanța între arbori pe baza metodei delimitatorilor

Algoritmul constă în retranscrierea structurii inițiale de tip arbore într-una de tip șir de primitive, folosind metoda delimitatorilor. Distanța se calculează prin algoritmul Wagner-Fisher clasic. Exemplificarea metodei se face folosind aceeași aplicație prezentată anterior.

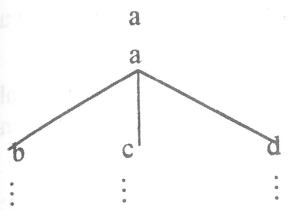
Ideea acestei prime metode de transformare arbore-șir este de a folosi două simboluri delimitatoare, "*(*" și "*)*", pentru a izola fiecare ramură a subarborilor din arborele analizat. Este evident că, pentru a putea realiza procesul de clasificare a șirurilor obținute în urma transformării, cele două caractere folosite la delimitare trebuie adăugate la mulțimea primitivelor deja existente.

Metoda de transformare este un proces iterativ, care construiește șirul prelucrând nodurile arborelui unul câte unul. Parcurgerea nodurilor arborelui se face în in-ordine, respectiv pornind de la rădăcină către extremități și de la stânga la dreapta. Pentru fiecare nod se construiește șirul după cum urmează:

a) dacă un nod are un singur descendent, atunci cele două primitive părinte și copil se trec concatenate în șir;

b) dacă un nod are mai mulți descendenți, atunci el este rădăcina unui subarbore și prin urmare, în șir, primitiva sa va fi urmată de scrierea între paranteze a ramurilor ce pornesc din el, în ordine de la stânga la dreapta.

Pentru exemplificare vom presupune următorul subarbore cu rădăcina a:



Porțiunea de șir obținut pentru acest subarbore are următoarea structură:

$$\ldots \quad a \quad (b\ \alpha) \quad (c\beta) \quad (d\ \gamma)\ldots \quad \cdots\cdots$$

	rădăcină subarbore	ramura din extrema stângă		Ramura din extrema dreaptă	

în care α, β și γ sunt structuri asemănătoare cu cea de sus, corespunzătoare subarborilor cu rădăcina b, c și respectiv d.

În urma aplicării metodei delimitatorilor pentru transformarea arborilor în șiruri, formele de intrare au următoarele descrieri:

A1: (a (a) (d)) (c c)
A2: (b (b) (d)) (c c)
A3: (b (b) (d)) (c b)
A4: (a (a) (d)) (b b)
A5: a (a) (d c)

C1: (a b d f) (d)
C2: (a b d f) (d)
C3: (a b e) (c)
C4: (a b d e) (b)
C5: (a b d) (d)

E1: (b (b d d) (d)) (d)
E2: (b (b d d) (d)) (d d)
E3: (a (b d d) (d)) (h) (d)
E4: (a (b d) (d)) (d)
E5: a (b d) (d) (d)

În tabelul 3, sunt calculate valorile medii ale distanţelor pentru forme din aceeaşi clasă şi, respectiv, din clase diferite.

Tabel 3. Valorile medii ale distanţelor în interiorul fiecărei clase şi între perechi de clase folosind algoritmul bazat pe metoda de transformare arbore-şir a delimitatorilor

	Clasa "A"	Clasa "C"	Clasa "E"
Clasa "A"	3.2	6.4	6.24
Clasa "C"	6.4	1.2	7.16
Clasa "E"	6.24	7.16	4.16

2.2. Distanţa între arbori pe baza metodei numărului de succesori

De această dată, păstrând algoritmul de calculare a distanţei între şiruri (Wagner-Fisher), se aplică pentru transformarea arborilor în şiruri metoda numărului de succesori.

Această nouă metodă de transformare arbore-şir păstrează informaţia structurală a arborelui prin intermediul numărului de succesori ai fiecărui nod. Aceste numere care se inserează în şirul de caractere vor face parte din mulţimea primitivelor, în procesul de clasificare a şirurilor.

În principiu, această metodă constă în parcurgerea arborelui de la stânga la dreapta, de la rădăcină către extremităţi, şi construirea şirului de primitive prin adăugarea progresivă a etichetelor arcelor care ajung în nodul curent, urmate de numărul de succesori ai acestora.

Prin aplicarea acestei metode numărului de succesori pentru transformarea arborilor în şiruri, structurile de date echivalente formelor din figura 1 sunt:

A1: 2 a 2 c 1 a 0 d 0 c 0
A2: 2 b 2 c 1 b 0 d 0 c 0
A3: 2 b 2 c 1 b 0 d 0 c 0
A4: 2 a 2 b 1 a 0 d 0 b 0
A5: 1 a 2 a 0 d 1 c 0

C1: 2 a 1 d 0 b 1 d 1 f 0
C2: 2 a 1 d 0 b 1 d 1 f 0
C3: 2 a 1 c 0 b 1 e 0
C4: 2 a 1 b 0 b 1 d 1 e 0
C5: 2 a 1 d 0 b 1 d 0

E1: 2 b 2 d 0 b 1 d 0 d 1 d 0
E2: 2 b 2 d 1 b 1 d 0 d 0 d 1 d 0
E3: 3 a 2 h 0 d 0 b 1 d 0 d 1 d 0
E4: 2 a 2 d 0 b 1 d 0 d 0
E5: 1 a 2 b 1 d 0 d 0

Tabel 4. Valorile medii ale distanţelor în interiorul claselor şi între perechi de clase în cazul aplicării metodei bazată pe transformarea arbore-şir a numărului de succesori

	Clasa "A"	Clasa "C"	Clasa "E"
Clasa "A"	2.64	5.48	7.6
Clasa "C"	5.48	1.92	7.08
Clasa "E"	7.6	7.08	5.28

2.3. Distanța dintre arbori pe baza metodei delimitatorilorși a distanței normate între șiruri

Această metodă este o îmbunătățire a celei prezentate în 2.1, atât timp cât folosindu-se aceeși metodă de transformare a arborilor în șiruri de primitive, respectiv metoda delimitatorilor, se încearcă o optimizare prin calcularea distanței între șiruri cu algoritmul Wagner-Fisher în variantă normată.

Folosind această metodă de transformare a arborilor în șiruri este evident că structurile de date folosite pentru acest algoritm sunt aceleași cu cele din 2.1.

Prin calcularea distanțelor normate între șiruri se obțin valorile distanțelor medii din tabelul 5.

Tabel 5 Valorile medii ale distanțelor în interiorul fiecărei clase și între perechi de clase folosind transformarea arbore=șir cu delimitatori și distanța Wagner-Fisher normată

	Clasa "A"	Clasa "C"	Clasa "E"
Clasa "A"	0.25	0.53	0.44
Clasa "C"	0.53	0.13	0.52
Clasa "E"	0.44	0.52	0.27

3. Concluzii

Pentru o mai bună comparare a metodelor prezentate, în tabelul 6 s-au folosit o serie de parametri ce caracterizează calitatea clasificarii formelor, prin intermediul cărora se realizează o prezentare mult mai concentrată a performanțele acestor algoritmi.

Tabel 6 Comparație între parametrii de caracterizare a calității clasificării

Parametrii comparați	Metoda directă (Lu)	Met. delimit. Dist. clasica	Met. nr. succ. dist. clasic	Met. delimit. dist. normată
Distanța medie intraclasă	2.1	2.85	3.28	0.21
Distanța medie interclase	3.02	6.6	6.72	0.5
distanta medie interclase / distanta medie intraclasa	1.43	2.31	2.05	2.38
distanta maxima intraclasa / distanta minima interclase	0.66	0.66	0.96	0.61
Timp de procesare a unei forme [secunde]	0.22	0.06	0.06	0.06

Se poate observa din acest tabel că rezultatele metodelor indirecte sunt mai bune decât cele obținute prin metoda directă a lui Lu, iar dintre metodele indirecte, la metoda care folosește distanța Wagner - Fisher normată se constată o ușoară superioritate.

Raportul (distanță medie interclase/distanță medie intraclasă) este un parametru ce caracterizează separabilitatea între clase. Aceasta este cu atât mai mare cu cât formele sunt mai concentrate în jurul mediilor, deci, o distanță medie intraclasă mai mică și, respectiv, cu cât distanța între clase este mai mare, ceea ce este echivalent cu o valoare mai mare a raportului (distanță medie interclase/distanță medie intraclase). Din tabelul 6 se observă că valoarea acestui parametru este de 1,43 pentru algoritmul lui Lu, iar pentru metodele indirecte este mai mare de 2, ajungând la valoarea maximă 2,38 în cazul metodei care folosește distanța normată și transformarea arbore-șir cu delimitatori.

Un alt parametru analizat este raportul (distanță maximă intraclasă/distanță minimă interclase), care, în esență, caracterizează calitatea clasificării pentru formele aflate la granița dintre clase. Algoritmul este mai bun cu cât acest parametru este mai mic, deoarece este evident că cea mai mare distanță între formele aceleiași clase

trebuie să fie cât mai mică, iar cea mai mică distanţă între forme din clase diferite trebuie să fie cât mai mare. Dacă acest raport ia valoarea 1 sau mai mare ca 1, rezultă că există forme în lot pentru care distanţa între formele aceleiaşi clase devine egală sau respectiv mai mare decât distanţa între forme din clase diferite, ceea ce înseamnă că apar clasificări greşite. Însă problema nu este atât de gravă cât timp raportul rămâne apropiat de valoarea 1, deoarece acest parametru caracterizează valorile extreme ale repartiţiei distanţelor între forme.

Din punct de vedere al acestui parametru se poate constata o uşoară superioritate a algoritmului bazat pe distanţa normată şi transformarea cu delimitatori (0,61) faţă de ceilalţi algoritmi, pentru care s-au obţinut 0,66 şi, respectiv, 0,96.

Ultimul parametru analizat este timpul necesar calculării respectivelor distanţe. Trebuie constatat iarăşi că metodele indirecte sunt mai bune, respectiv timpul este de 0,06 secunde faţă de 0,22 secunde necesare prelucrării în cazul algoritmului direct al lui Lu.

Bibliografie

1. **ANREWS, H. C.**: Introduction to Mathematical Techniques in Pattern Recognition, Wiley, New York, 1972.
2. **ATANASIU, I., MATEESCU, A.**: Limbaje formale, Bucureşti, 1990.
3. **BUNKE, H.**: Attributed Programmed Graph Grammars and Their Application to Schematic Diagram Interpretation to Texture Analysis. În: I.E.E.E. Trans. Pattern Anal. Mach. Intell., vol. PAMI -4, no. 6, Nov. 1982, pp. 1103-1114.
4. **ESHERA, M.A., FU, K. S.**: A Graph Distance Measure for Image Analysis. În: I.E.E.E. Trans. Syst., Man, Cybern., vol. SMC-13, no. 3, March 1994, pp. 398-408.
5. **EVANS, T.G.**: Grammatical Inference Techniques in Pattern Analysis. În: Software Engineering, vol. 2, J. T. Tou (Ed.), Academic Press, New York, 1971.
6. **FILIPSKI, A. J.**: A Least Mean - Squared Error Approach to Syntactic Classification", I.E.E.E. Trans. Pattern Anal. Mach. Intell., vol. PAMI - 2, no. 3, March 1980, pp. 252-255.
7. **GRIGORE, O.**: Syntactical Clustering Using the Guided Random Search Method. În: A Special Issue of Buletinul Ştiinţific al Universităţii "Politehnica" Timişoara and Trans. Automatic Control and Computer Science and Engineering, Nov. 1996, pp. 174-182.
8. **GRIGORE, O.**: Syntactical Clustering Using Genetic Algorithms. În: Real World Applications on Intelligence Technologies, Ed. Academiei Române, 1996.
9. **GRIGORE, O.**: Syntactical Self - Organization Map, Computational Intelligence, Theory and Applications, B. Reusch (Ed.), Springer, Dortmund 1997, pp. 101-109.
10. **GRIGORE, O.**: The Cumulative Inference algorithm for Regular Grammars. În: Proc. of the 10-th Scandinavian Conference on Image Analysis, June 1997, Finland, pp. 191-199.
11. **GONZALEZ, R. C., WINTZ, P.**: Digital Image Processing, Addison - Wesley, MA 1996.
12. **LAM, L., LEE, S. W., SUEN, S. Y.**: Thinning Methodologies - A Comprehensive Survey. În: I.E.E.E. Trans. Pattern Anal. Mach. Intell., Vol. PAMI-14, No. 9, Sept. 1992, pp. 869-885.
13. **LEMONE, K. A.**: Similarity Measures Between Strings Extended to Sets of Strings", I.E.E.E. Trans. Pattern Anal. Mach. Intell., vol. PAMI-4, no. 3, May 1982, pp. 345-347.
14. **LU, S. Y., FU, K. S.**: Error - Correcting Tree Automata for Syntactic Pattern Recognition and Image Processing", RPI, Troy, NY 1977, pp. 115-127.
15. **LU, S. Y., FU, K. S.:** A Sentence - to - Sentence Clustering Procedure for Pattern Analysis. În: I.E.E.E. Trans. Syst., Man, Cybern., vol. SMC-8, no. 5, May 1994, pp. 219-227.
16. **LU, Y.**: A Tree- to -Tree Distance and Its Application to Cluster Analysis. În: I.E.E.E. Trans. Pattern Anal. Mach. Intell., vol. PAMI - 1, no. 2 April 1979.
17. **LU, Y.**: A Tree- to -Tree Distance and Its Application to Cluster Analysis. În: I.E.E.E. Trans. Pattern Anal. Mach. Intell., vol. PAMI - 1, no. 2 April 1979.
18. **NEAGOE, V.-E., STĂNĂŞILĂ, O.**: Teoria recunoa]terii formelor, Ed. Academiei Române, Bucureşti, 1992.
19. **PAVLIDIS, T.**: Structural Pattern Recognition, Springer - Verlag, Berlin, 1977.
20. **WAGNER, R. A., FISHER, M. J.**: The String to String Correction Problem. În: J. Ass. Comput. Mach., vol. 21, no. 1, Jan. 1974, pp. 42-51.