

# CODIFICAREA INFORMAȚIEI ÎN NOILE MODELE DE CALCUL. CALCUL BAZAT PE ADN ȘI CALCULUL CU MEMBRANE CELULARE

LUCIANA MARIA MOROGAN

*Universitatea Spiru Haret*

**Rezumat:** Cercetări recente au considerat atât ADN-ul, cât și membranele celulare, drept medii pentru calculul la scară mare dar și pentru stocarea ultra compactă a informației (ADN-ul acționează drept mecanism de stocare și codificare a informației genetice, deși nu este folosit direct în catalizarea proceselor celulare, ci, mai degrabă, ARN-ul și proteinele acționează drept enzime ce catalizează reacțiile din interiorul celulelor). Lucrarea de față însumează un colaj de modele de codificare a informației (colaj realizat în urma cercetării printre lucrările scrise până în prezent) privite din perspectiva a două noi modele de calcul inspirate de micro – biologie: calculul bazat pe ADN și calculul folosind P-sisteme (realizat pentru reprezentarea unară a unui număr întreg pozitiv scris sub formă binară).

**Cuvinte cheie:** DNA computing, membrane computing, P systems.

**Clasificare AMS:** 94A60, 92D99, 74K15

## 1. INTRODUCERE

Tratamentul matematic al geneticii reprezintă un subiect vechi, cu o tradiție onorabilă ce ne duce înapoi la Gregor Mendel, cercetător austriac care a demonstrat în 1865 cum caracteristicile plantei de mază au fost moștenite de la o generație la alta într-un mod previzibil. Recent însă, conexiunea dintre matematică și biologie a devenit mai apropiată în urma analizării cantității de date biologice, aflate într-o continuă creștere, pe baza conceptelor matematice fundamentale, ceea ce face parte din *“revoluția biologică, cea mai mare revoluție științifică a timpurilor noastre, și, posibil, a tuturor timpurilor”* (Michael C. Reed – matematician de renume).

Compresia și transmiterea datelor nu este aplicabilă doar mediilor electronice: natura a “dezvoltat” un sistem capabil să ofere codajul instrucțiunilor necesare supraviețuirii celulare. Poate este corect să afirmăm că cel mai important cod existent

este *codul genetic*, dar, să nu ometem faptul că acesta reprezintă, în esență, doar un cod. Din aceste considerente, o aplicație o reprezintă *sistemele criptografice moleculare* (ce folosesc codificarea informației). În stilul altor ramuri ale *calculului natural* (*Natural Computing*), învățăm de la natura vie noi modele de calcul și noi strategii (să le spunem... paradigme) bazate pe *calculul ADN-ului* (*DNA computing*) și pe și mai noul model de calcul: *calculul cu membrane* (*membrane computing*) – *P sisteme* (încercări timide se cunosc însă în aceste două direcții).

## 2. REMARCI GENERALE

O *codificare* este o aplicație injectivă ce asociază fiecărui cuvânt sursă dintr-o mulțime  $X$ , un șir de simboluri ale unui alfabet  $Y$ . Presupunem că avem cuvintele sursă  $x_i, x_j$  cu  $i \neq j$  și  $f(x_i) \neq f(x_j)$ . Un *mesaj* se definește ca fiind orice șir de cuvinte sursă din  $X = \{x_1, \dots, x_m\}$ . Numim *cuvinte cod* acele șiruri peste  $Y$  de forma  $f(x_i), 1 \leq i \leq m$ . Mulțimea de cuvinte cod de forma  $f(x_i)$  reprezintă un *cod*  $C$ . Operația de codificare  $f$  se poate extinde natural la mulțimea mesajelor:  $f(xy) = f(x)f(y)$ .

O codificare  $f$  se numește a fi *unic decriptabilă* dacă nu există două mesaje  $M_1 \neq M_2$  astfel încât  $f(M_1) = f(M_2)$ . O codificare  $f$  este un *cod instantaneu* (*cod prefix*) dacă nu există două cuvinte sursă  $x_i \neq x_j$  astfel încât  $f(x_i)$  să fie prefix pentru  $f(x_j)$  (nu putem avea  $f(x_j) = f(x_i)y, y \in Y$ ). Astfel, un cod prefix poate fi codificat în mod unic, de la stânga la dreapta, fără a “privi înainte”. Codurile prefix sunt cunoscute și sub numele de *coduri libere de prefix* (*prefix-free codes*) [13].

**Lemă.** Dacă  $f$  este un cod liber de prefix, atunci codificarea  $f$  este unic decriptabilă. Demonstrația lemei se găsește în [13].

Considerăm  $q$  un număr prim și  $n \in \mathbb{N}^*$ . Se numește *cod liniar* orice subspațiu liniar al lui  $Z_q^n$ . Un subspațiu  $k$  dimensional al lui  $Z_q^n$  se numește  $(n, k)$  – *cod liniar* peste alfabetul  $Z_q$  și se notează  $A_{n,q}$ . Acesta admite o bază formată din  $k$  vectori cu  $n$  elemente. Dacă  $A_{n,q}$  este un cod liniar cu baza  $e_1, \dots, e_k$ , atunci matricea

$$G_{n,k} = \begin{pmatrix} e_1 \\ \vdots \\ e_k \end{pmatrix}$$
 se numește matricea generatoare a codului. Operația de codificare,  $\Phi$ ,

este definită prin matricea generatoare:  $\forall u \in Z_q^k, \Phi(u) = uG$ . Dacă  $A_{n,q}$  este un cod liniar generat de matricea  $G = G_{n,k}$ , se numește *matrice de control* o matrice  $H = H_{n-k,n}$  cu proprietatea ca  $GH^T = 0$  de rang  $n - k$ .

*Distanța Hamming*  $H(x,y)$  dintre două cuvinte  $x = x_1x_2\dots x_n$  și  $y = y_1y_2\dots y_n$ , reprezintă numărul indicilor  $i$  astfel încât  $x_i \neq y_i$  (*distanța Hamming* a unui cod liniar

este cea mai mică distanță dintre elementele codului). Spunem că un  $(n, k)$  - cod liniar are distanța minimă  $d$  dacă și numai dacă poate detecta orice combinație de maxim  $d - 1$  erori (dacă și numai dacă poate corecta orice combinație de maxim  $\left\lfloor \frac{d-1}{2} \right\rfloor$  erori) și verifică relația  $d \leq n - k + 1$ .

Se numește *extensie* a unui  $(n, k)$  - cod liniar  $A$  peste  $Z_q$ ,  $(n + 1, k)$  - codul liniar  $A^*$  obținut prin adăugarea la fiecare cuvânt cod  $a_1, \dots, a_n$  a unui simbol nou  $a_{n+1}$  (în cazul binar numit *bit de paritate*) cu proprietatea că  $\sum_{i=1}^{n+1} a_i = 0 \pmod{q}$ . Pentru  $H$  matricea de control a codului  $A_{n,q}$ , considerăm matricea de control  $H^*$  a codului extins  $A_{n+1,k}^*$  ca fiind de forma

$$H^* = \left( \begin{array}{cccc|c} & & & & 0 \\ & & & & \vdots \\ & & & & 0 \\ \hline 1 & 1 & \dots & 1 & 1 \end{array} \right)$$

Dacă un cod liniar binar  $A$  are distanța minimă impară  $d$ , atunci codul extins are distanța minimă  $d + 1$ . Ultima linie a matricii de control reprezintă ecuația:  $\sum_{i=1}^n x_i = 1$ . Un cod  $A$  de lungime  $n$  corectează  $t$  erori și detectează  $s$  erori simultan dacă orice cuvânt cod  $v$  are proprietatea:

$$\forall w \in Z_q^n [d(w, v) \leq s \Leftrightarrow \forall a \in A \setminus \{v\}, d(w, a) > s].$$

**Teoremă.** Un cod corectează  $t$  erori și detectează  $s$  erori simultan dacă și numai dacă  $d \geq t + s + 1$  [13].

Codul liniar binar în care coloanele matricii  $H$  sunt reprezentarea binară a numerelor  $1, 2, \dots, 2^r - 1$  se numește *cod Hamming binar*. Deci, pentru orice  $r \in \mathbb{N}$  ( $r \geq 2$ ) putem construi un  $(n, k)$  - cod liniar în care  $n = 2^r - 1, k = 2^r - r - 1$ . Cum un cod  $H$  are distanța minimă 3, poate corecta o eroare și detecta 2 (dar nu poate realiza simultan acest lucru). Codul  $H$  poate fi îmbunătățit prin extensie, operație ce duce la un  $(2^m, 2^m - m - 1)$  - cod liniar cu toate cuvintele cod de pondere pară. Un cod  $H$  extins este soluția unui sistem liniar de  $n - k + 1$  ecuații; ecuația suplimentară

$\sum_{i=1}^{n+1} x_i = 0$  se numește *ecuația de control a parității*. Un cod  $H$  extins are  $d = 4$ . Codul  $H$  extins corectează o eroare simplă și detectează două erori simultan.

Considerăm un cod de lungime  $n = n_1 n_2$ . Cuvintele codului pot fi reprezentate ca matrici cu  $n_1$  linii și  $n_2$  coloane prin scrierea cuvântului cod  $\mathbf{a} = a_0 a_1 \dots a_{n-1}$  ca o

matrice  $X = [x_{i,j}]$ ,  $0 \leq i \leq n_1 - 1$ ,  $0 \leq j \leq n_2 - 1$ , unde  $x_{i,j} = a_{in_2+j}$ , numită *scriere canonică*. Dacă  $A_1, A_2$  sunt două coduri liniare de lungime  $n_1$  respectiv  $n_2$ , putem construi codul  $A$  de lungime  $n_1 n_2$  folosind scrierea canonică. Spunem că  $A = A_1 \times A_2$  este *codul produs* al lui  $A_1$  cu  $A_2$  dacă și numai dacă  $A$  constă din toate cuvintele – cod în care reprezentarea matriceală verifică proprietățile: (i) Fiecare coloană a unei matrici este un cuvânt – cod din  $A_1$ ; (ii) Fiecare linie a unei matrici este un cuvânt – cod din  $A_2$ .

**Structura ADN-ului.** ADN-ul (*Acidul DeoxiriboNucleic*) [17] este un *polimer* (substanță compusă din molecule cu masă moleculară mare, alcătuite din unități structurale repetate sau monomeri repetați, conectați prin legături chimice covalente) alcătuit din *monomeri* (cuvânt de proveniență grecească, *mono* “unu” și *meros* “parte” - moleculă mică ce poate fi legată chimic de alți monomeri alcătuind astfel un polimer) numiți *deoxiribonucleotide*. ADN-ul reprezintă molecula crucială a organismelor vii, având o structură fascinantă care suportă cele mai importante două funcții ale acesteia: codarea pentru producerea proteinelor și propria reproducere astfel încât o copie exactă a acesteia să fie trecută noilor celule ce urmează a fi create.

Speranțele înalte privind viitorul calculului ADN-ului se bazează pe două trăsături fundamentale [1]: (i) *paralelismul* masiv al catenelor de ADN și (ii) *complementaritatea* Watson-Crick. Descriem pe scurt cele două trăsături.

- (i) Majoritatea faimoaselor probleme de calcul se rezolvă printr-o căutare exhaustivă prin toate soluțiile posibile. Dificultatea incontestabilă constă în faptul că o asemenea căutare este prea vastă pentru a putea fi dusă la capăt folosind tehnologia actuală. Pe de altă parte, densitatea informației, conținută în moleculele de ADN, cât și ușurința de construcție a copiilor acestei informații pot face posibile aceste căutări exhaustive. Un exemplu îl constituie criptanaliza, într-un sistem clasic de criptare (cu chei relativ puține – până la  $2^{56}$ ), unui text cifrat în care toate posibilele chei pot fi încercate simultan [1].
- (ii) Complementaritatea Watson-Crick reprezintă o trăsătură furnizată “gratuit” de către natură. Când au loc legăturile dintre cele două catene ale moleculei de ADN, știm că bazele opuse una celeilalte sunt complementare. Dacă cunoaștem unul dintre membrii legăturii, automat îl cunoaștem și pe celălalt.

**Structura membranei.** *Calculul cu membrane (Membrane Computing - MC)* ([2] și [3]) este o latură a informaticii bazată pe idei de calcul abstract și modele ale structurii și funcționalității celulelor vii, cât și pe modul în care celulele sunt organizate în țesuturi sau alte structuri superioare ordonate. Dispozitivul de calcul cu membrane este numit *P sistem*. Ingredientul principal al unui P sistem îl reprezintă *structura de membrană* reprezentată de o mulțime de membrane așezate ierarhic conținute într-o membrană externă distinctă, care corespunde plasmei membranei

celulare, numită *piele (skin)*. În interiorul acesteia pot fi plasate mai multe membrane, fiecare determinând un compartiment numit *regiune*. Compartimentele unei celule conțin substanțe ce înoată într-o soluție apoasă. Aici nu există ordine, orice este închis la orice, ceea ce contează este doar concentrația, adică populația, *numărul de copii ale fiecărei molecule*. Aceasta sugerează lucrul cu mulțimi de obiecte ale căror multiplicitate contează, numite *multiseturi*. Formal, fie  $U$  o mulțime arbitrară, un *multiset* peste  $U$  este o aplicație  $M : U \rightarrow N$ ,  $M(a)$  reprezentând multiplicitatea lui  $a$  în multisetul  $M$ , oricare ar fi  $a$  din  $U$ . Putem indica aceasta prin perechea  $(a, M(a))$ .

Obiectele evoluează prin intermediul unor *reguli de evoluție* a căror localizare este asociată regiunilor unei celule. Există trei tipuri de astfel de reguli: reguli de rescriere multiset (membrane și reguli de evoluție), reguli de comunicare și reguli de dezvoltare([2], [3]).

### 3. SCRIEREA INFORMAȚIEI ÎN ADN

O dată cu tehnologia care avansează, cererea pentru scrierea informației în interiorul ADN-ului crește. Domeniile de aplicabilitate imediată, asemenea biotehnologiilor convenționale în sensul că încearcă să *codifice informația artificială în ADN*, sunt:

1. *calculul bazat pe ADN* (care încearcă să realizeze matematica biologică, adică, rezolvarea problemelor matematice prin aplicarea metodelor experimentale în biologia moleculară; aplicații se găsesc și în domeniul criptografiei);
2. *memorarea datelor de tip ADN* (care advocă folosirea ADN-ului bacterial ca structură de memorare a datelor de densitate mare și de lungă durată și care poate fi rezistent la radiații);
3. *semnătura ADN-ului* (care este importantă pentru înregistrarea “copyright”-ului genomului bacterial și a celui viral). Steganografia (semnătură invizibilă ascunsă în altă informație) este folositoare în acest sens.

#### 3.1. CERINȚE PENTRU CREEAREA UNUI COD DE TIP ADN

Considerăm o mulțime de cuvinte de tip ADN pentru schimbul de informație. Cuvintele trebuie să fie distincte și de lungimi egale, astfel încât nici un cuvânt să nu producă hibridizări nedorite în ceea ce privește aranjamentul acestora. (*Hibridizarea* reprezintă procesul de combinare a acizilor nucleici monocatenari complementari într-o singură moleculă. Nucleodidele se vor lega la complementele lor în condiții normale, proces numit *renaturare*[17]). În același timp, toate cuvintele trebuie să fie uniforme, din punct de vedere fizico-chimic, pentru a garanta, în experimentele biologice, o reacție în care probabilitatea de emisie a valorii 1 este de exact  $\frac{1}{2}$  (termenul folosit în limba engleză pentru o secvență de acest tip este cel de *unbiased* [11]).

În principiu, există două măsuri pentru evaluarea calității cuvintelor ADN create: statistic termodinamice și combinatoriale. Deși metodele termodinamice redau

o estimare destul de clară, costul de calcul este destul de ridicat, motiv pentru care ne axăm pe estimările combinatoriale, care sunt în esență apropiate de cele termodinamice. În cele ce urmează vor fi introduse cerințele formale pentru mulțimea de cuvinte de tip ADN.

### 3.1.1. CONSTRÂNGERI ASUPRA SECVENȚELOR

Fie  $x = x_1x_2\dots x_n$  un cuvânt ADN generat de cele patru baze  $\{A, C, G, T\}$  ({adenine, citosine, guanine, timine}). *Inversul* lui  $x$  este  $x^R = x_nx_{n-1}\dots x_1$  iar *complementul* lui  $x$ , obținut prin înlocuirea bazelor A cu T și C cu G în  $x$  și vice-versa, îl notăm cu  $x^C$ . Pentru o mulțime  $S$  de cuvinte ADN, notăm complementarea cu secvențele complementului opus cu

$$S^{RC} = \{x \mid x \in S \text{ sau } (x^R)^C \in S\}. \quad (1)$$

**Constrângeri Hamming** [4]. La fel ca în teoria codurilor, cuvintele ADN proiectate trebuie să păstreze o distanță Hamming mare între perechile de cuvinte. Ceea ce face proiectarea codului ADN-ului mai complicată decât teoria standard a codurilor corectoare de erori (a se vedea bibliografia [11], [12] și [13]) este faptul că trebuie să considerăm nu doar  $H(x, y)$  ci și  $H(x^C, y^R)$  pentru a garanta *nepotrivirile* hibridizării cu alte cuvinte și complementele acestora. (O bază necomplementară într-o catenă dublă nu poate forma legături de hidrogen stabile și poartă numele de *nepotrivire* (a bazei). În engleză, (base) *mismatch* [13]).

**Constrângeri de tip “liber de prefix”** [4]. În proiectarea cuvintelor ADN se dorește ca acestea să fie de tipul “libere de prefix”. Proprietatea prin care orice cuvânt concatenat diferă de orice alt cuvânt prin cel puțin  $d$  poziții se numește *liber de prefix de index d*. Deci, codul ADN trebuie să fie de tipul “liber de prefix” cu index mare.

**Constrângeri legate de energie** [4]. Temperatura de topire a tuturor cuvintelor ADN trebuie să fie asemănătoare pentru a garanta comportamentul lor comun *in vitro*. Estimarea pe care ne putem baza este aproximarea cea mai apropiată la care temperatura trebuie calculată de la frecvența de 16 baze dimer (de la AA la TT), unde *frecvența dimer* a unei secvențe  $x$  este reprezentată de tripletul de întregi ce reprezintă, fiecare, frecvențele a trei șabloane. Pentru a integra și bazele terminale se presupune că  $x$  este ciclic în calculul frecvenței. Arita și Kobayashi [18] au propus această aproximare grupând [GC] și [AT] ale căror temperaturi de topire depind de frecvența a trei șabloane: [GC][GC], [GC][AT] sau [AT][GC] și [AT][AT].

**Alte constrângeri** [4] ce depind de modelul folosit sunt:

1. *Subcuvintele interzise* ce corespund locurilor de restricție, repetărilor simple sau altor secvențe de semnal biologice, nu trebuie să apară oriunde în cuvintele proiectate. Aceste constrângeri apar în cazul în care modelul de

codificare folosește secvențe pre -determinate cum ar fi locurile de restricție pentru enzime.

2. *Orice cuvânt de lungime  $k$*  nu trebuie să apară mai mult de o dată în cuvântul dorit. Numărul  $k$  este de obicei mai mare sau egal cu 6.
3. *O structură secundară* ce implică hibridizarea așteptată a cuvintelor de tip ADN nu trebuie să existe. Pentru a găsi structura optimă pentru aceste cuvinte, minimum de energie liberă a catenelor este calculat de programarea dinamică. Această constrângere apare în cazul în care controlul temperaturii este important pentru modelul de codificare.
4. *Doar trei baze, A, C și T* pot fi folosite în proiectarea cuvintelor. Această constrângere servește în principal la reducerea numărului de nepotriviri. Pentru cuvintele proiectate pe baza ARN-ului, constrângerea este importantă pentru stabilitatea ambelor perechi  $G - C$  și  $G - U$  (echivalentul perechii  $G - T$  în ADN).

### 3.1.2. MODURI DE STOCARE A DATELOR

**Abordarea bazată pe suprafață (faza solidă)** [4] presupune plasarea cuvintelor ADN pe un suport solid. Cum una dintre cele două catene ale unei molecule dublu catenare este imobilizată, cuvintele cod pot fi separate de complementele lor, reducând astfel riscul unei agregări neașteptate a cuvintelor – ceea ce reprezintă un avantaj al utilizării acestei metode. Un alt avantaj îl reprezintă eficacitatea etichetării fluorescente, deoarece este mai simplu de recunoscut cuvintele pentru citirea informației.

**Abordarea solubilă (faza lichidă)** [4]. Accesul la informație din cazul anterior poate limita abilitățile biomoleculare. Fragmente de ADN în soluție sunt capabile să simuleze automate celulare drept purtători ai informației, datorită flexibilității acestor fragmente. Un alt avantaj al abordării solubile îl constituie posibilitatea introducerii de ADN în microbi și astfel, cuvintele să poată fi folosite pentru proiectarea unor nano-structuri.

*În concluzie*, problema proiectării este formulată astfel: fiind dați doi întregi  $l$  și  $d$  ( $0 < d < l$ ), se poate proiecta o mulțime  $S$  de lungime  $l$  de cuvinte ADN astfel încât  $S^{RC}$  este liberă de prefix de index  $d$  și pentru orice două secvențe  $x, y \in S^{RC}$ ,

$$H(x, y) \geq d$$

și

$$H(x^C, y^D) \geq d.$$

### 3.2. METODE DE PROIECTARE A CODULUI DE CUVINTE ADN

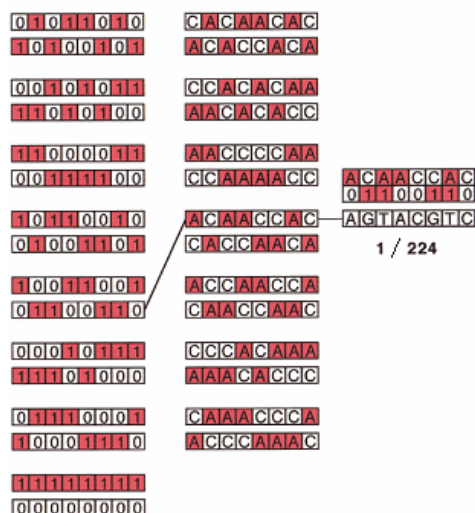
**Strategia șablon-plan.** Tratamentul matematic al generării oligonucleotidelor se bazează pe strategia șablon-plan, strategie ce utilizează codurile Hamming [13] pentru a oferi șabloane și plane necesare generării de secvențe ADN. Un șablon

reprezintă o secvență nucleotidică asupra a doua baze, în timp ce un plan reprezintă un șir de variabile binare. Secvențele de cuvinte ADN sunt generate de operația dintre fiecare plan, al unei mulțimi date, asupra fiecărui șablon din altă mulțime.

O mulțime de molecule de lungime  $l$ , în care toate distanțele Hamming sunt mai mari sau egale cu  $n$ , se notează ca mulțime de forma  $l:n$ . Considerând acestea, strategia funcționează bine pentru cazul  $l = 4k$  și  $n = 2k$ , cu  $k$  număr întreg ( $4k:2k$ ). Mulțimile de cuvinte ADN pot fi generate și pentru cazurile  $l \neq 4k$  în care strategia este folosită pentru crearea de mulțimi de tipul:  $(4k-1) : (2k-1)$ ,  $(4k+1) : 2k$  și  $(4k-2) : (2k-1)$ . În lucrarea de față prezentăm doar cazul  $4k:2k$  (pentru mai multe detalii, a se urmări [19]).

În funcție de numărul de plane și șabloane, pot fi generate mulțimi de oligonucleotide de diferite dimensiuni. Fiecare pereche șablon-plan ( $t, m$ ) generează o secvență ce satisface regula: pentru fiecare bit 1 al planului, bitul corespunzător din șablon se schimbă cu complementul său, și pentru fiecare bit 0, bitul corespunzător din șablon rămâne neschimbat. De exemplu, o pereche șablon-plan de forma (ACAACCAA, 01100110) se folosește pentru generarea următoarei secvențe: AGTACGTA (a se vedea figura 1).

16 plane X 14 șabloane = 224 secvențe



**Figura 1.** Mulțimea de plane și șabloane folosite pentru a genera mulțimea de tip  $l:n = 8:4$ . Fiecare monomer de lungime 8 este generat prin încrucișarea fiecărui șablon cu fiecare plan, după regula menționată.

În acest caz, mulțimea de plane și șabloane folosite pentru generarea mulțimii de tip  $l:n = 8:4$  conduce la generarea unei secvențe de 224 4bm 8 meri (224 monomeri, de lungime 8, cu distanță Hamming 4 – prescurtarea “bm” provenind de la expresia în limba engleză “base mismatch”). Fiecare plan are o distanță Hamming de cel puțin 4 (diferă prin cel puțin 4 poziții de celelalte plane).



Pentru determinarea mulțimii de plane,  $M$ , și a mulțimii de șabloane,  $T$ , codului Hamming  $i$  se aplică două constrângeri: (i) conținutul  $G/C$  al fiecărei mulțimi este fixat la aproximativ 50% pentru asigurarea unei stabilități termodinamice similare pentru fiecare catenă dublă de ADN, și (ii) orice două oligonucleotide de lungime  $l$  din mulțime diferă printr-o distanță Hamming de cel puțin  $n$  ( $n \approx l/2$ ). (Considerând parametrul distanței  $n$ , pentru orice pereche de cuvinte distincte  $(x, y)$  din mulțime, trebuie ca  $H(x, y) \geq n$ ). Fiecare secvență de ADN se distinge în mod unic de celelalte secvențe prin hibridizarea complementului acesteia.

Deci considerăm astfel, codul Hamming binar de forma  $(l, G, n)$  ca fiind o mulțime de vectori  $G$  peste  $\{0, 1\}$ , de lungime  $l$ , astfel încât orice doi vectori dețin între ei o distanță Hamming de cel puțin  $n$ . Pentru generarea unui număr maxim de secvențe pentru mulțimi de tip  $4k:2k$ ,  $k$  număr întreg, se impun două condiții: (i) mulțimea planelor,  $M$ , să fie reprezentată de codurile Hamming  $(l, 2l, l/2)$  și (ii) mulțimea șabloanelor,  $T$ , să fie similară mulțimii  $M$ , mai puțin vectorii ce conțin doar 0 și cel ce conține doar 1, și, în plus, înlocuim  $\{0, 1\}$  din codul Hamming cu perechi de baze, cum ar fi:  $\{A, C\}$ ,  $\{A, G\}$ ,  $\{T, C\}$ , sau  $\{T, G\}$  și vice versa. În exemplul de mai sus,  $\{0, 1\}$  a fost înlocuit cu  $\{C, A\}$  pentru generarea mulțimii de șabloane  $T$ .

**Cod ADN corector de erori cu constrângeri libere de prefix.** Printre diferitele constrângeri asupra cuvintelor de cod ADN, cea mai dificilă de satisfăcut este cea de tipul “liber de prefix”[4]. Această proprietate reprezintă o condiție necesară pentru proiectarea generală a unui cod ADN. Prezentăm în continuare un cod ADN de 112 cuvinte de lungime 12 având distanța Hamming mai mare sau egală cu 4 în orice hibridizare nedorită (greșită). Cuvintele nu conțin mai mult de 6 subsecvențe consecutive comune. Temperaturile de topire ale cuvintelor sunt similare.

În proiectare este implicată metoda lui Arita și Komayashi [14] care generează sistematic o mulțime de cuvinte de lungime  $l$  astfel încât oricare dintre membrii săi va avea o distanță Hamming de aproximativ  $l/3$  în raport cu alte cuvinte și cu complementele lor, și, în plus, se suprapun cu concatenările acestora. La fel ca în strategia șablon-plan, s-au construit secvențe ca produs de două tipuri de cuvinte binare, cu excepția folosirii unui singur cuvânt binar,  $T$ , ca șablon.  $T$  este ales astfel încât la alinierea cu orice model, să dețină o distanță Hamming mai mare sau egală cu  $d$ :

$$T^R \quad TT^R \quad T^R T \quad TT \quad T^R T^R$$

Șablonul specifică pozițiile  $GC$  ale cuvântului dezvoltat:  $[GC]$  va corespunde în șablon fie lui 0 fie lui 1. Cum  $T^R$  specifică șablonul  $AT/GC$  de complemente inverse, distanța Hamming dintre  $T$  și  $T^R$  garantează distanța Hamming a bazelor din catenele următoare și din cele inverse ale ADN-urilor ce urmează a fi dezvoltate (proiectate). Alte șabloane de la  $TT$  la  $T^R T^R$  sunt responsabile de șabloanele shiftate (șabloane deplasate la stânga sau la dreapta cu un număr de biți).

Pentru cuvintele de tip plan, se folosește orice cod binar corector de erori de distanță minimă  $d$ . Astfel, orice pereche de cuvinte a codului rezultat induce o distanță Hamming de cel puțin  $d$ , fără deplasări bloc de biți datorită codului corector

de erori și cu deplasări bloc datorită șablonului ales. Cum un singur șablon este folosit pentru specificarea pozițiilor  $GC$  pentru toate cuvintele, aranjamentul  $GC$  al codului rezultat va fi uniform și, deci, vor rezulta temperaturi de topire similare pentru toate cuvintele.

#### 4. CODIFICAREA INFORMAȚIEI SUB FORMĂ DE NUMERE BINARE ÎN STRUCTURA MEMBRANEI FOLOSIND MULTISETURI

Prezentăm în continuare o metodă simplă de codificare a numerelor binare folosind multiseturi (multiplicarea), și o familie de  $P$  sisteme ce transformă aceste multiseturi în notație unară. Ideea codificării numerelor folosind multiseturi se poate extinde la codificarea informației.

Fie  $x_n x_{n-1} \dots x_1$  reprezentarea binară a unui întreg pozitiv  $x$  astfel încât

$$x = \sum_{i=1}^n x_i 2^{i-1}. \quad (2)$$

Vom folosi obiecte din următorul alfabet:

$$A_n = \{ \langle b, j \rangle \mid b \in \{0, 1\}, j \in \{1, 2, \dots, n\} \},$$

unde obiectul  $\langle b, j \rangle$  reprezintă bitul  $b$  de pe poziția  $j$  în codificarea binară a unui număr întreg.

Pentru reprezentarea numărului  $x$  de mai sus, folosim multisetul :

$$\langle x_n, n \rangle, \langle x_{n-1}, n-1 \rangle, \dots, \langle x_1, 1 \rangle \quad (3)$$

(alfabetul  $A_n$  depinde de lungimea reprezentării binare a lui  $x$  și deci folosind  $A_n$  putem reprezenta numere întregi din intervalul cuprins între 0 și  $2^n - 1$ )

Pe de altă parte, reprezentarea unară a lui  $x$  se obține prin alegerea unui simbol  $a$  dintr-un alfabet  $A'$  și punând în multiset  $x$  copii ale acestui simbol:  $a^x$ . În concluzie, reprezentarea unară este exponențial mai lungă decât cea binară. Astfel, pentru a oferi valorile de intrare pentru  $P$  sisteme, trebuie inserat în acestea un număr exponențial de obiecte ceea ce duce la o cantitate exponențială de lucru pentru pregătirea sistemului. În schimb, lucrul cu numere codificate binar, permite prepararea sistemului prin inserarea unui număr de obiecte limitat polinomial.

##### 4.1. REPREZENTAREA UNARĂ A NUMERELOR BINARE

În cele ce urmează, este prezentată o familie de  $P$  sisteme ce permit reprezentarea, în notație unară, a unui număr întreg pozitiv dat  $x$ , exprimat în notație binară. Obiectele folosite de  $P$  sistemele din familie formează o submulțime a alfabetului  $A_n$  definit mai sus. Pentru reprezentarea lui  $x$  în notație binară se folosesc doar obiecte care corespund biților egali cu 1 ai reprezentării lui  $x$ . *De exemplu*, dacă  $x = 23$ , vom avea reprezentarea binară a lui  $x$  prin 10111 și vom folosi obiectele:  $\langle 1,$

$5\rangle$ ,  $\langle 1, 3\rangle$ ,  $\langle 1, 2\rangle$  și  $\langle 1, 1\rangle$ . Cum primul element al perechilor este întotdeauna 1, îl vom omite și implicit vom omite și parantezele unghiulare.

Familia P sistemelor care va produce transformarea se definește astfel:

$$\Pi = (A(n), \mu, w, R(n), i_{in}, i_{out}), \text{ unde:}$$

- $A(n) = \{1, 2, \dots, m\} \cup \{a\}$  este alfabetul;
- $\mu = []_{skin}$  reprezintă structura membranei ce conține doar membrana skin;
- $w = \emptyset$  - multisetul inițial de obiecte prezente în regiunea marcată de membrană;
- $R(n)$  – mulțimea regulilor de evoluție care sunt de forma:

$$[j \rightarrow (j-1)^2]_{skin}, \text{ pentru } \text{toti } j \in \{1, 2, \dots, n\}$$

$$[j \rightarrow a]_{skin}$$

- $i_{in} = skin$  specifică membrana de intrare a lui  $\Pi(n)$ ;
- $i_{out} = skin$  specifică membrana de ieșire a lui  $\Pi(n)$ .

Semantica regulilor este semantica obișnuită a regulilor de evoluție (a se vedea bibliografia, [3] și [6]) aplicate într-o manieră paralelă. Numărul pașilor celulari ai P sistemului este limitat de  $n$  și calculul se oprește când nici o altă regulă nu mai poate fi aplicată, iar multisetul plasat în membrana de ieșire reprezintă output-ul calculului.

**Realizarea calcului.** În general, un *calcul* al unui P sistem începe plecând de la o configurație inițială a sistemului și se termină atunci când nici o regulă de evoluție nu mai poate fi aplicată. Rezultatul va reprezenta multisetul de obiecte conținut în *membrana de ieșire* sau eliberat din membrana skin a sistemului în mediul înconjurător.

În cazul nostru, inițial, în regiunea delimitată de membrana skin sunt introduse obiecte specificând pozițiile lui 1 din reprezentarea binară a lui  $x$ . Calculul începe și regulile din  $R(n)$  sunt aplicate. Se observă că prezența obiectului  $j$  ( $j \in \{1, 2, \dots, n\}$ ) va produce  $2^{j-1}$  copii ale obiectului  $a$ . Deci, la sfârșitul calculului (când regulile din  $R(n)$  nu se mai pot aplica), membrana skin va conține  $x$  copii ale obiectului  $a$ , adică exact reprezentarea unară a lui  $x$ . Un exemplu de calcul va fi ilustrat în continuare, considerând, ca mai sus, pe 23 ca numărul a cărui reprezentare binară se dorește.

Pentru  $x = 23$ ,  $x$  va fi reprezentat de obiectele 5, 3, 2 și 1 (copii unice).

$1 \rightarrow a$   
 $2 \rightarrow 1, 1$   
 $3 \rightarrow 2, 2$   
 $5 \rightarrow 4, 4$

Pas1: se aplică în paralel regulile și rezultă multisetul  $a, 1, 1, 2, 2, 4, 4$ .

$1 \rightarrow a$   
 $1 \rightarrow a$   
 $2 \rightarrow 1, 1$   
 $2 \rightarrow 1, 1$   
 $4 \rightarrow 3, 3$   
 $4 \rightarrow 3, 3$

Pas2: se aplică în paralel regulile și rezultă multisetul  $a, a, a, 1, 1, 1, 1, 3, 3, 3, 3 = a^3, 1^4, 3^4$ .

$1 \rightarrow a$   
 $3 \rightarrow 2, 2$

Pas3: se aplică în paralel regulile și rezultă multisetul  $a^3, a^4, 2^8 = a^7, 2^8$ .

Pas4: se aplică în paralel regulile  $2 \rightarrow 1, 1$  și rezultă multisetul  $a^7, 1^{16}$ .

Pas5: se aplică în paralel regula  $1 \rightarrow a$  și rezultă multisetul  $a^{23}$ .

## 5. CONCLUZII

Natura vie oferă idei fundamentale pentru noi modele de calcul. Lucrarea de față prezintă două strategii de codificare a informației în stilul *calculului natural*. Direcțiile abordate se bazează scrierea informației în ADN și pe proiectarea unui cod ADN, dar și pe mai noul model de calcul cu membrane celulare (P sistemele).

**Paradox:** asemănătoare și totuși atât de diferite! În ceea ce privește implementarea modelului de calcul cu membrane, nu este clar, la acest moment, dacă trebuie să ne orientăm spre biochimie – cum este cazul calculului bazat pe ADN, sau spre computerul electronic (în sensul celor create în scop general, sau a celor special create în acest sens) – cum este cazul rețelelor neurale și a algoritmilor genetici.

În modelul de bază al calculului cu membrane, nu avem de-a face cu șiruri și limbaje (așa cum găsim în calculul bazat pe ADN), ci cu *multiseturi* de *obiecte atomice*. Iată unde intervin diferențe între cele două abordări ale codificării informației: nu sintaxa este cea care contează aici (vezi calculul bazat pe ADN - [1]), ci vectorii numerici care caracterizează multiplicitatea copiilor obiectelor ce aparțin unei anumite regiuni. Iată, deci, și de ce apare un asemenea paradox : deși sunt două abordări inspirate din natură, sunt totuși atât de diferite! Atunci, teoria P-sistemelor

reprezintă o ramură a... ? *Teoria multiseturilor formale sau teoria limbajelor formale*, ca în cazul calculului bazat pe ADN?

Fundamental P sistemelor, aplicate codificării numerelor folosind multiseturi, nu este numai lucrul cu aceste multiseturi, ci și *structura membranei*, împreună cu toate consecințele acesteia (în special posibilitatea lucrului în *regiuni* separate, aranjate ierarhic, și *comunicarea* dintre regiuni; este important să subliniem că membranele sunt *separatori* și *canale de comunicație*) ducând la posibilitatea extinderii codificării numerelor la codificarea informației.

Calculul cu membrane provine din biologie, unde, împreună cu domeniul biochimiei, procesele sunt nedeterministe, rezultatul fiind doar aproximativ/adevărat din punct de vedere probabilistic. Până acum, în domeniul P sistemelor, a fost folosită doar matematica “pură”, ceea ce este de altfel adevărat și în domeniul calculului ADN-ului. Dar ce putem spune despre abordările matematice “aproximative”, folosind probabilități? Ce putem spune despre calculul “aproximativ”, orice ar însemna aceasta?

Toate acestea pot fi reformulate în termeni de *soft computing* care, se pare, aparțin *calculului molecular* (calcul bazat pe ADN și calcul cu membrane) [2].

## BIBLIOGRAFIE

1. Gh. Paun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*, Ed. Springer-Verlag Berlin Heidelberg, 1998.
2. Gh. Paun, *Computing with membranes (P systems): Twenty six research topics*,
3. C. S. Calude, M.J. Dinneen, Gh. Paun, *Pre-proceedings of the workshop on multiset processing (WMP-CdeA 2000)*, pages 203-217, CDMTCS-140, August 2000.
4. C. Calude, Gh. Paun, *Computing with cells and atoms*, Taylor and Francis, London, 2000 (Chapter 3: *Computing with membranes*).
5. N. Jonoska, Gh. Paun, G. Rozenberg, *Aspects of molecular computing, A festschrift in honor of the 70<sup>th</sup> birthday of Tom Head*, Ed. Springer-Verlag Berlin Heidelberg, 2003. (Chapter 2: M. Arita, *Writing information into DNA*)
6. N. Jonoska, Gh. Paun, G. Rozenberg, *Aspects of molecular computing, A festschrift in honor of the 70<sup>th</sup> birthday of Tom Head*, Ed. Springer-Verlag Berlin Heidelberg, 2003 (Chapter 12: A. Gehani, T. LaBean, J. Reif, *DNA-based cryptography*)
7. G. Ciobanu, Gh. Paun, M. J. Perez-Jimenez, *Applications of membrane computing*, 2005, format electronic.(Chapter 1: Gh. Paun, *Introduction to membrane computing*)
8. M. A. Gutierrez-Naranjo, A. Leporati, C. Zandron, *Converting integer numbers from binary to unary notation with P systems*, disponibil pe <http://psystems.disco.unimib.it>
9. A. Alhazov, C. Bonchis, G. Ciobanu, C. Izbasa, *Encodings and Arithmetic Operations in P systems*, disponibil pe <http://psystems.disco.unimib.it>
10. G. Michaels, *Cryptography and linguistic of macromolecules, Studying the codes and language of life*, EMSL Lecture notes, April 2004.
11. A. Gehani, T. LaBean, J. Reif, *DNA-based cryptography*, 5<sup>th</sup> annual DIMACS meeting on DNA based computers (DNA 5), MIT, Cambridge, MA, June 1999.

12. Aiden A. Bruen, Mario A. Forcinito, *Cryptography, Information Theory, and Error-Correction: a handbook for the 21<sup>st</sup> century*, Wiley-Interscience, John Wiley&Sons, Inc., Hoboken, New Jersey.
13. A. Atanasiu, *Criptografie, Note de Curs*, Disponibil pe: [http://www.galaxyng.com/adrian\\_atanasiu/cript.htm](http://www.galaxyng.com/adrian_atanasiu/cript.htm)
14. A. Atanasiu, *Teoria codurilor corectoare de erori*, Ed. Universitații București, 2001.
15. A. Atanasiu, *P - systems and Arithmetic Calculus*, Research Group on Mathematical Linguistics, Universitat Rovira i Virgili Tarragona (Spania) 14/00 (2000).
16. A. Atanasiu, *Arithmetic with membranes*, ROMJIST 4 (2001), nr. 1-2, pp. 5-20.
17. Atanasiu, C. Martin – Vide, *Recursive Calculus with membranes*, Fundamenta Informaticae 49 (2001), 1-15.
18. Dr. A. M. Israil, *Biologie Moleculară. Prezent și Perspective*, Ed. Humanitas București, 2000.
19. M.Arita, S.Kobayashi, *DNA Sequence Design Using Templates*, New Generation Comput. 20(3), 263277(2002). lucrare disponibilă pe <http://www.ohmsha.co.jp/ngc/index.htm>.
20. M. Li, H. J. Lee, A. E. Condon, R. M. Corn, *DNA Word Design Strategy for Creating Sets of Non-interacting Oligonucleotides for DNA Microarrays*, Langmuir 18(3), 805812(2002).
21. <http://psystems.disco.unimib.it>
22. [[www.dcs.shef.ac.uk](http://www.dcs.shef.ac.uk)]

**Abstract:** Researchers considered the DNA and the cell membranes as devices for the storage of genetic material. Recently, they implemented the storage of huge quantities of information in DNA and more recently, in some abstract devices called membranes systems, known as well as P systems – called after their inventor, prof. dr. Gh. Păun. This paper presents the necessary terminology of understanding the new concepts in section 2, called “general remarks”. Those two different categories: of writing the information into DNA and of coding the information into the membrane structure using multisets of atomic objects, are presented in sections 3 and 4, respectively. In section 3 are presented the solid and the liquid faze of writing the information in the DNA sequences along with the constraints necessary for such processes. A totally different approach is illustrated in the next chapter, because in the base model we don't have to deal with sequences, as in DNA computing, but with multisets of atomic objects, as we have already explained. This is where the differences between two nature based aproches, of coding the information, actually differ. What really matters here, is not the syntax of the membrane model, who totally differ of the DNA based model, but the numerical vectors, that characterized the multiplicity of object copies belonging of certains regions. Some of those conclusions are presented in the last section of this paper.