Mobile Networks and Applications (2023) 28:499–506
https://doi.org/10.1007/s11036-022-02058-7

An Encrypted Proposal Method in Membrane Computing Aggregation (MCA)

Alberto Arteta Albert[1] · **Yanjun Zhao**[1] · **Luis Fernando de Mingo López**[2] · **Nuria Gómez Blas**[2]

Accepted: 25 October 2022 / Published online: 9 November 2022

Abstract
MCA (MCA (Membrane Computing Aggregation is an experimental bioinspired computational frame, inspired by the inner properties of membrane cells. It is capable of problem-solving activities by maintaining a special, meaningful relationship with the internal/external environment, integrating its self-reproduction processes within the information flow of incoming and outgoing signals. The amount of information that can process varies and it is supposed to enhance performance as per the parallel processing properties found in Nature. The paper describes a way of encrypting the information an MCA handle and also introduces an application of such a system. The results can be used as a building block for using complex cryptographic properties in alternative and emerging computational models.

Keywords Membrane computing aggregation · Encrypted P-systems · Natural computing

1 Introduction

Nature-inspired computing is no longer and experimental field. Recent advances in bioinspired systems are a reality [2, 3]. Despite all the recent emphasis and advancements in systems biology, synthetic biology, and network science about modeling gene networks, protein networks, metabolic and signaling networks, etc. some of the most important computational properties of membrane cells have not been grappled and abstracted et: scalability, tissular differentiation, and morphogenesis -i.e., the capability to informationally transcend the cellular level and organize higher-level information processes utilizing heterogeneous populations of membrane cells organized as computational tissues and organs. Currently, modules include switches, cascades, pulse generators, oscillators, spatial patterns, and logic formulas [32].The second wave of synthetic biology is integrating basic parts and modules to create systems-level circuitry. genomes and synthetic life organisms are envisioned, and application-oriented systems are contemplated. Different computational tools and programming abstractions are actively developed (the Registry of Standard Biological Parts; the Growing Point Language GLP; the Origami Shape Language OSL, the PROTO bio programming language, etc. See details at the Open Wetware site). Evolving cell models of prokaryotes have also been addressed [11, 14]. As some have put it, systems broaden the scope of synthetic biology by designing synthetic circuits to operate reliably in the context of differentiating and morphologically complex membrane cells presenting unique challenges and opportunities for progress in the field [21]. However, very few synthetic biology researchers do contemplate using systems. In systems biology, a plethora of modeling developments has been built around signaling pathways, cell cycle control, topologies of protein networks, transcriptional networks, etc.

Alberto Arteta Albert, Yanjun Zhao, Luis Fernando de Mingo López and Nuria Gómez Blas contributed equally to this work.

✉ Alberto Arteta Albert
aarteta@troy.edu

Yanjun Zhao
yjzhao@troy.edu

Luis Fernando de Mingo López
fernando.demingo@upm.es

Nuria Gómez Blas
nuria.gomez.blas@upm.es

[1] College of Arts and Sciences, Troy University, 129-A MSCX, 600 University Avenue, Troy 36082, Alabama, USA

[2] Departamento de Sistemas Informáticos, ETS Ingeniería de Sistemas Informáticos, Universidad Politécnica de Madrid, Alan Turing, Madrid 28031, Madrid, Spain

There is relatively well-consolidated thinking, in part due to traditional physiology and systems science and control theory which was at the origins of this new field, of going from genes to membrane cells to the whole organ as Denis Noble has done for heart models [28]. The integration of proteins into organs has also been promoted by bioinformatic-related projects such as the Physiome Project. [24]. Important works have been done in the vicinity of network science to make sense of gene networks, protein networks, transcription networks, complex formation, etc. For instance, about how is dynamically organized modularity in the yeast protein-protein interaction network [20], it was uncovered that two types of hubs contribute to the organized modularity of the proteome: party hubs which interact with their partners simultaneously, and date hubs, which bind their different partners at different times and locations (we will see later on the importance of the discussion on modularity in the evo-devo field). Predictive models of mammalian membrane cells have been described using graph theory, assembling networks, and integrative procedures [27]. Important systems biology compilations and far-reaching cellular models have been made by [10, 23, 26, 29]. It has to be emphasized that concerning the views advocated in this proposal, most systems biology works depart from the goal of abstracting computational power out from systems and focus instead on applying computational power to analyze the organization of systems. Notwithstanding the foregoing, studies such as [16] on bacteria as computers, and [33] on the operating system of bacteria could be considered forerunners in the former direction. In [9] an example of MCA is introduced showing P-systems communication based on the principles of [31]. In [4] A minimal biological unit was measured using MCA principles.

The communication errors and correction protocols are enhanced by the encryption procedures described in the next sections.

In the science of development (the evo-devo discipline) most of the emphasis has been on modularity. What it exactly means in developmental terms is still a matter of controversy [13, 15, 34]; but undoubtedly modularity refers to the capability of cellular networks to dissociate networked processes at a lower level and to recombine or redeploy them at the higher level of the multicellular organism. Thanks to the cellular signaling system, the genetic switches, the cytoskeleton, and some other topo-biological mechanisms, the unitary network of cellular processes integrated into the cell cycle may be broken down into coherent modules and be performed separately in different membrane cells within differently specialized tissues [30]. This implies a flexible organization for the deployment of biomolecular processing modules, which are cut differently in each tissue along the developmental process, due also to chromatin remodeling during development [22]. Interestingly, not only differentiation but also morphology becomes an instance of the scalable modular processing, throughout the tensegrity emergent property and the ontogenetic arrangement of symmetry breakings in a force field. The emergence of cellular bauplan where signaling, force fields, and cytoskeletal mechanical modes work to create basic morphologies for membrane cells, depending also on the populations. Interestingly, complex morphologies obtained from the Turing diffusion model have been cogently discussed as a result of cell-to-cell developmental interactions [25]. Currently, the evo-devo field accumulates a considerable mass of biomolecular-organization-facts, poorly conceptualized yet, to be computationally abstracted in the perspective of MCA advancement.

In the fields closer to computer science and Biocomputing, it has been important the introduction of the agent-based approach (as pioneered by W. Fontana and others), which uses sets of rules to define relationships between cellular components substituting for the simple Boolean networks and differential equations used up to now. Proteins and other biomolecules become molecular automata and the aggregate behavior that emerges from these models is the combinatorial expression of all those automata doing specific micro-functions [12]. This approach shows promise for the evolvable advancement of network models endowed with the flexible modularity property. It is somehow close to the already mentioned predictive models of mammalian membrane cells that are using graph theory, assembling networks and integrative procedures [27]. New generations of cellular models (of automata) have been developed too, with powerful data content and with potential for modeling multi-cellular systems in a general way, supporting user-friendly in silicon experimentation and discovery of emergent properties [5]. Under the approach of Artificial Embryology, a developmental system has been obtained using of cellular automata systems capable of following rewriting rules procedures, emulating elementary morphologies, and multicellular distributions [17].

As for the developments in molecular Biocomputing, the idea that biomolecules (DNA, RNA, proteins) might be used for computing already emerged in the fifties and was reconsidered periodically with more and more arguments which made it more viable. But the definitive confirmation came in 1994 [1]. Regarding applicative models, there are many attempts to update Cells computing paradigm [6–8, 18, 19] among others. However, there is no record of an aggregated system that uses encrypted information. This proposal integrates a codification method that allows adding cybersecurity to P-systems, and in particular MCA.

2 Membrane computing

A Transition P System of degree n, with $n > 1$, is a construct

$$\Pi = (V, \nu, \omega_1, \cdots, \omega_n, (R_1, \rho_1), \cdots, (R_n, \rho_n), i_0) \tag{1}$$

where:

- V is an alphabet (its elements are called objects);
- ν is a membrane structure of degree n, with the membranes and the regions labeled in a one-to-one manner with elements in a given set; in this section, we always use the labels 1, 2, n;
- ω_j, with $1 \leq j \leq n$, are strings from V^* representing multisets over V associated with the regions $1, 2, \cdots, n$ of nu;
- R_i, with $1 \leq i \leq n$, are a finite set of evolution rules over V associated with the regions $1, 2, \cdots, n$ of ν;
- ρ_i is a partial order over R_i, $1 \leq i \leq n$, specifying a priority relation among rules of R_i. An evolution rule is a pair (u, v) which we will usually write in the form $u \rightarrow v$ where u is a string over V and $v = v'$ or $v = v'\delta$ where v' is a string over $(V \times \{here, out\}) \cup (V \times \{in_j \ 1 \leq j \leq n\})$ and δ is a special symbol not in. "here" means for the object to stay in the same region and "out" means for the object to leave the region. The length of u is called the radius of the rule $u \rightarrow v$;
- i_0 is a number between 1 and n which specifies the output membrane of Π (Fig. 1).

Definition 1 Multiset of Objects. Let U be a finite and not empty set of objects and N the set of natural numbers. A multiset of objects is defined as a mapping:

$$\begin{aligned} M: \quad & U \rightarrow N \\ & a_i \rightarrow u_i \end{aligned} \tag{2}$$

, where $a_i \in U$ s an object and $u_i \in N$ its multiplicity.

As it is well known, there are several representations for multisets of objects.

$$M = \{(a_1, u_1), (a_2, u_2), \cdots, (a_n, u_n)\} = a_1^{u_1} a_2^{u_2} \cdots a_n^{u_n} \tag{3}$$

Note that the initial Multiset is the multiset existing within a given region in which no application of evolution rules has occurred yet.

Definition 2 Evolution rule with objects in U and targets in T s defined by $r = (m, c, \delta)$ where $m \in M(U), c \in M(U \times T)$ and $\delta \in \{todissolve, nottodissolve\}$ (from now on c will be referred as the consequent of the evolution rule r).

Note that the set of evolution rules with objects in U and targets in T is represented by $R(U, T)$. Targets are multisets or membrane actions

Definition 3 Multiplicity of an object in a multiset of objects $M(U)$. Let $a_i \in U$ be an object and let $m \in M(U)$ be a multiset of objects. The multiplicity of an object is defined over a multiset of objects such as:

$$\begin{aligned} a_i: \quad & U \times M(U) \rightarrow N \\ & (a_i, m) \rightarrow m_{ai} = n | (a_i, n) \in m \end{aligned} \tag{4}$$

Definition 4 Multiplicity of an object in an evolution rule r. Let $a_i \in U$ be an object and let $R(U, T)$ be a multiset of evolution rules. Let $r = (m, c, \delta) \in R(U, T)$ where $m \in M(U), c \in M(U \times T)$ and $\delta \in \{todissolve, nottodissolve\}$. The multiplicity of an object is defined over an evolution rule such as:

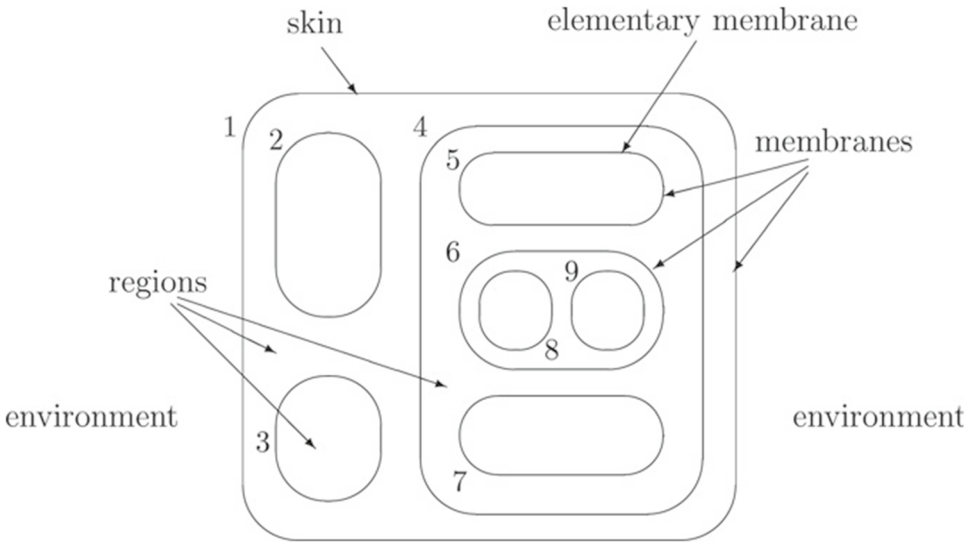

Fig. 1 P-System structure

$$\begin{aligned} a_i : \quad & U \times R(U,T) \to N \\ & (a_i, r) \to m_{ai} = n | (a_i, n) \in m \end{aligned} \tag{5}$$

Definition 5 P-system evolution. Let C_i be the consequent of the evolution rule r_i. Thus, the representation of the evolution rules is:

$$\begin{aligned} r_1 &: a_1^{u_{11}} a_2^{u_{12}} \cdots a_n^{u_{1n}} \to C_1 \\ r_2 &: a_1^{u_{21}} a_2^{u_{22}} \cdots a_n^{u_{2n}} \to C_2 \\ & \cdots \\ r_m &: a_1^{u_{m1}} a_2^{u_{m2}} \cdots a_n^{u_{mn}} \to C_m \end{aligned} \tag{6}$$

P-systems evolve, which makes them change over time; therefore, it is a dynamic system. Every time that there is a change in the p-system the p-system is in a new transition. The step from one transition to another one will be referred to as an evolutionary step, and the set of all evolutionary steps will be named computation. Processes within the p-system will be acting in a massively parallel and nondeterministic manner. (Similar to the way the living cells process and combine information).

3 Membrane computing aggregation

A Membrane Computing Aggregation (MCA) is a set $\Omega = \{\Pi_0, \Pi_1, \cdots, \Pi_N\}$ and a set Φ of aggregation rules among membranes. The set of aggregation rules is not fully integrated with the evolution rules of a given p-System but establishes the correlation between 2 given membrane models by deciding the way 2 or more P-systems are being aggregated. The rules can be defined as a matrix relation:

$$\varphi_1(k_1, k_2, \cdots, k_m) = \begin{pmatrix} u_{11} & u_{21} & \cdots & u_{m1} \\ u_{12} & u_{22} & \cdots & u_{m2} \\ \cdots & \cdots & \cdots & \cdots \\ u_{1n} & u_{2n} & \cdots & u_{mn} \end{pmatrix} \begin{pmatrix} k_1 \\ k_2 \\ \cdots \\ k_m \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \\ \cdots \\ u_n \end{pmatrix} \tag{7}$$

where $\varphi_1(\bar{k})$ is the aggregation relation and is defined by the association of n P-systems, k determines the aggregation rules of each component in every P-system I and U are the component (objects).

The evolution rule application phase in every region within a P-system, is described as follows:

The application of rules to a multiset of objects in a region is a transforming process that has an input, an output and the conditions for transforming the P-system. Given a region within a P-system, let$U = \{a_i \ 1 \le \mathrm{i} \le n\}$ be the alphabet of objects, m a multiset of objects over U, and $R(U, T)$ a multiset of evolution rules with antecedents in U and targets in T. The input in the region is the initial multiset m.

The output is a maximal multiset m'. The transformations have been made based on the application of the evolution rules over m until m' is obtained. Application of evolution rules in each region of P-systems involves subtracting objects from the initial multiset by using rules antecedents. Rules used are chosen in a nondeterministic manner. This phase ends when no rule is applicable anymore. The transformation only needs rules antecedents as the consequents are part of the communication phase.

Let $k_i \in N$ be the number of times that the rule r_i is applied. Therefore, the number of symbols a_j which have been consumed after applying the evolution rules a specific number of times will be:

$$\sum_{i=1}^{m} k_i u_{ij} \tag{8}$$

Definition 6 Given a region R and alphabet of objects U, and $R(U, T)$ set of evolution rules over U and targets in T.

$$\begin{aligned} r_1 &: a_1^{u_{11}} a_2^{u_{12}} \cdots a_n^{u_{1n}} \to C_1 \\ r_2 &: a_1^{u_{21}} a_2^{u_{22}} \cdots a_n^{u_{2n}} \to C_2 \\ & \cdots \\ r_m &: a_1^{u_{m1}} a_2^{u_{m2}} \cdots a_n^{u_{mn}} \to C_m \end{aligned} \tag{9}$$

A maximal multiset is that one that complies with:

$$I_{l=1}^{m} \left(Y_{i=1}^{n} \left(u_i - \sum_{j=1}^{m} \left(k_j u_{ji} \right) \le u_{li} \right) \right) \tag{10}$$

The following sections focus on the encryption schemes, adding one example for clarity.

4 Encryption stage

Manipulating chars is more complex than numbers. That is why for each char, a number is linked to its code. Then operations applied to that number are easier to perform. The symbol m has a unique identifier.

$f(m)$: for the encryption method, we use x as a generic object and a bijective function. (m (string) belongs to V (alphabet)). $f(m)$ returns a number x that is sent over a channel. For this proposal, we have chosen standard symmetric encryption based on a monoalphabetic substitution adding a light load of computational complexity. As MCA models are experimental, more complex cipher algorithms such as DES,3DES or AES will be good candidates when MCA becomes more stable. The communication phase in MCA works as follows:

The receiver obtains $x = f(m)$ and it applies $f^{-1} \rightarrow f^{-1}(f(m)) = m$ and then the receiver obtains the string (m) This occurs in the best-case scenario. Unfortunately, the transmission might alter a few bits of the sent information.

In that case, the encryption method catches the error as the substitution uses a bijective function to do the mapping of the plaintext/ciphertext. If an error occurs, the decryption phase will not be able to decrypt properly the original word sent across the regions and the P-system will display an error.

$$f : A \rightarrow V | m \in A \tag{11}$$

$$f(m) = y \in V, m = f^{-1}(y) \in A \tag{12}$$

For each symbol m of the alphabet it searches the symbol $y = f(m)$ numeric representative. (A number) This number y is sent over the channel and the symbol y' is received.

- If no noise is made, then $y' = f(m)$ and the receiver gets m by applying the inverse of $f : f^{-1}$.
- If the noise in the channel produces an error in the transmission then y' will be received. If f^{-1} returns a number of V then the receiver considers that there is no error although the symbol sent was a different one. Note: We will work mainly with vector spaces Z_2^n.

Vector $(x_1, x_2, \cdots, x_n) \in Z_n$. That is linked to the string $x_1 x_2 \cdots x_n, x_i \in \{0, 1\}$.

4.1 Example and simulation

Given the alphabet $A = \{a, b, c, d, e, f, g, h\}$ with 8 elements. The identifiers are generated by the function below: $f : A \rightarrow Z_8$.

This function codifies the characters of A with number between $\{0, \cdots, 7\}$. It is the position in alphabetical order $a = 1, b = 2, c = 3, \cdots, h = 7$. As f is biyective the alphabet A can be identified by a set of strings in Z_2^3. In binary code $0 = 000, 1 = 001, 2 = 010, 3 = 011, \cdots$.

So, $Z_2^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

The MCA consists of an aggregation of 2 P-systems with 3 regions each. Every region has 1 multiset m=ab and the same evolution rules: ($a \rightarrow b, b \rightarrow out$) To find the best fit for our example the codification for ab=001. We send the word $y = f(m) = 001$ over the communication channel. The noise either alters it or not. If the noise does not alter the word, the receiver gets the original word (001 = 1). And then, by calculating $f^{-1}(1)$ we obtain the initial word a. What would happen if the noise alters the communication (for example the first bit of the number 001). In this case, the receiver gets $y' = 101$ and it does not detect an error because the number codified as 101 also exists. That is a problem.

To avoid the mentioned problem, the decoding part consists of 2 different processes:

1. Detecting a transmission error through the channel and then finding out which is the word sent. That is why we use a new coding function called c so $y = c(f(m))$.
2. Obtaining the symbol m. Sometimes, the function c has properties that allow error detection.

Example 1 Let us suppose that the source is again $A = \{a, b, c, d, e, f, g, h\}$ and is related to $Z_2^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$. We consider a new function or application c that expand the bits of the word. It is defined as:

$$c(x_1, x_2, c_3) = (x_1, x_2, x_3, x_1 + x_2, x_1 + x_3, x_2 + x_3) \tag{13}$$

so $c(110) = (110011)$. The first 3 bits are the same and the rest changes. In other words, it expands the word 110 to 110001. In terms of vector spaces, c is defined as:

$$c : Z_2^3 \rightarrow Z_2^6 \tag{14}$$

It transforms the vector of Z_2^3 into another Z_2^6. Now there are 2 power 3 words:

$$C = \{000000, 001011, 010101, 011110, 100110, 101101, 110011, 111000\} \tag{15}$$

Now the words are different from each other in 3 bits. What happens now if the transmission process alters bits?

In that case, the noise has to alter 3 bits from the system and does not detect an error. If there are more than three. The system might not detect there is an error. For example, if we send the word $y = 001011$ and the noise alters its first bit, then 101011 is received. In this case, an error has occurred. It detects that something has gone wrong as the word 101011 does not exist in C. We define C as the set of words that have been expanded.

Let us suppose that the noise alters the first three bits of y, and the receiver gets the word $y'' = 110011$ that does belong to C. in this case the error is not detected and the decoding is wrong. Our focus are words that are far from each other. That is to say, words are very different in terms of the number of different bits. When that happens errors can be detected more easily along with the encryption control.

Technically we add more bits at the end of the word.

In our example, we add three bits. These last three bits are redundant. It means they do not contain information about the word sent. The info is used to control the possible errors that might occur. The last three bits act as a control process and allow the detection of errors, and in some cases, correction.

4.2 Coding functions

Definition 7 Linear code of length n: If the number of words is equal to k is said that C is a (n, k)-linear code and we can interpret that C is the code associated with a coding function $c : k$.

Z_p is the alphabet from numerous sources. The elements of C are called codewords. If C is a (n, k)-linear code is said that the

Definition 8 Redundancy r: $r = n - k$. In general, $n > k$ and $r > 0$. If C is a code over Z_2 is said to be binary. If C is a code over Z_3 is said to be ternary.

Example 2 Words have 6 bits. This means we work on Z_6. We consider the vector subspace $C = L(100110, 010101, 001011)$. Is verified that the number of independent words is 3. This implies $Dim(C) = 3$ and therefore $k = 3$.

Thus, C is a (6, 3)-linear binary code. In addition, C has redundancy $n - k = 6 - 3 = 3$. Redundancy means the number of bits used to control the error in transmission. Then, if we call G a matrix whose rows are the words of C

$$G = \begin{pmatrix} 1\ 0\ 0\ 1\ 1\ 0 \\ 0\ 1\ 0\ 1\ 0\ 1 \\ 0\ 0\ 1\ 0\ 1\ 1 \end{pmatrix} G^t = \begin{pmatrix} 1\ 0\ 0 \\ 0\ 1\ 1 \\ 0\ 0\ 1 \\ 1\ 1\ 0 \\ 1\ 0\ 1 \\ 0\ 1\ 1 \end{pmatrix} \tag{16}$$

and

$$G^t(x_1, x_2, x_3) = \begin{pmatrix} 1\ 0\ 0 \\ 0\ 1\ 1 \\ 0\ 0\ 1 \\ 1\ 1\ 0 \\ 1\ 0\ 1 \\ 0\ 1\ 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{17}$$

We call c to the function than codes any word by using G. Therefore, in this case:

$$c(x_1, x_2, c_3) = (x_1, x_2, x_3, x_1 + x_2, x_1 + x_3, x_2 + x_3) \tag{18}$$

For instance, what is the code for the word 101?

$C(1, 0, 1) = (1, 0, 1, 1, 0, 1)$. Note the first three bits are the same and three more bits are added (Control bits).

Definition 9 We define distance between words $u = (u_1, \cdots, u_n)$ and $v = (v_1, \cdots, v_n)$ as:

$$d(u, v) = card\{i | u_i \neq u_j, 1 \leq i \leq n\} \tag{19}$$

In other words, the number of different bits. Formally the definition is distance of code C:

$$d(C) = min\{d(u, v) | u, v \in C, u \neq v\} \tag{20}$$

For example, in a three dimensional space Z_2^3, $d(100, 000) = d(100, 010) = 1$, $d(100, 011) = 3$ i f f $C = \{000000, 001111, 010111, 011000, 100111, 101000, 110000, 111111\} \in Z_2^6$. Thus, $d(C) = 2$.

Here are some properties:

1. $d(u, v) = d(u, w) + d(w, v)$
2. $d(u + v) = d(u) + d(v)$
3. If C is a (n, k)-linear code then:
   - $d(u, v)$ is the weight of the words u, v which is the minimum number of 1's .
   - $d(C)$ is equal to the weight of C, that is, $d(C) = weight(C)$

To calculate the distance of a linear code, it is enough to find the weight of all the non-null words (number of 1's) and then return the minimum number. That number is the distance of C.

Example 3 Let $d(111, 011) = d(111, 001) + d(001, 011)$ and $d(111 + 011) = d(111) + d(011)$. Note how to calculate the distance of the following linear codes:

$$C = \{000000, 001111, 010111, 011000, 100111, 101000, 110000, 111111\} \tag{21}$$

$$C_2 = \{000000, 001011, 010101, 011110, 100110, 101101, 110011, 111000\} \tag{22}$$

If $d(C) = d$ then it is necessary at least to change d bits to convert one word into another. In the beginning of this chapter we studied the (6, 3)-linear code $C = L(100110, 010101, 001011)$.

If the word y' that is received is 101011 and it belongs to C, If the transmission channel is proven to be decrease the probability of finding error, the word sent originally is the closest possible word to C, that is to say the word that has the most number of bits equal to y'. If we calculate the distance from y' to all the words of C_2 we obtain:

$$d(y', 001011) = 1, d(y', 101101) = d(y', 110011) = 2, d(y', 100110) = 3 \tag{23}$$

$$d(y', 111000) = 3, d(y', 000000) = d(y', 011110) = 4, d(y', 010101) = 5 \tag{24}$$

Then, we decode y' as $u = 001011$ which belongs to C. If only one is found, then $u = y$. In this case we the error is nor only detected, but also corrected. This is a brief description of the decoding method by the minimal distance.

5 Decoding stage

C is a (n, k, d)-linear code. Let us suppose that we send the word y that belongs to C through a channel y. Because of the noise, the word y' is received.

If y' does not belong to C, then a transmission error is detected and we try to obtain the word y'.

To do that two decoding methods are introduced.

- Minimal distance method.
- Síndrome method

5.1 Minimal distance method

- Input: All the words of (n, k, d)-code, C and y' the word received.
- Output: y belongs to C, and y' (y belongs to C).

We calculate as $d(y, y') = min\{d(c, y')|c \in C\}$. The output is the closest word to y.

Example 4 Let $C = \{00011, 11001, 10101, 11111\}$ code that verifies $d(C) = 2$. We received these 2 words $y' = 01100$ and $y'' = 11110$. Using the minimal distance method.

- $d(01100, u)$ and u belongs to C with $C = \{00011, 11001, 10101, 11111\}$

$$d(01100, 00011) = 4, d(01100, 11001) = 2, d(01100, 10101) = 3, d(01100, 11111) = 3$$

- $d(11110, u)$ and u belongs to C with $C = \{00011, 11001, 10101, 11111\}$

$$d(11110, 00011) = 4, d(11110, 11001) = 3, d(11110, 10101) = 2, d(11110, 11111) = 1$$

If the number of words (n) is very high the decoding method is very complex because we would need to compare n couple of bits. If the code is less complex because by definition: $d(u, v) = d(u + v)$.

Example 5 Let $(6, 3, 3)$ be a linear code with

$$C = L(100110, 011110, 001011) = 000000, 100110, 011110, 001011, 010101, 101101, 110011, 111000$$

We received $y' = 111111$. It detects an error.

Minimal distance method:

- $d(111111, u) = w(111111 + u)$ where u belongs to C.

$$000000 = (111111 + 000000) = w(111111) = 6$$
$$100110 = (111111 + 100110) = w(011001) = 3$$
$$011110 = (111111 + 011110) = w(100001) = 2$$
$$001011 = (111111 + 001011) = w(110100) = 3$$
$$010101 = (111111 + 010101) = w(101010) = 3$$
$$101101 = (111111 + 101101) = w(010010) = 2$$
$$110011 = (111111 + 110011) = w(001100) = 2$$
$$111000 = (111111 + 111000) = w(000111) = 3$$

Thus, there are three possible ways to decode y' as there are three words with a distance equal to 2. This way of distance calculation is used because the code is linear. By doing so, the calculation costs are lower.

6 Conclusions

Encryption in Membrane computing is a new approach and an integrative tool to provide the transition P-Systems as structures that can handle coded information to be used in cybersecurity applications. Optimal applications could further consider implementations in deciphering and decoding as a part of RSA, DES, AES, or 3DES algorithms processing, for example. Since then, new variations have been suggested to try to fit this model to new realities. The main goal for this unconventional paradigm is to improve the performance of the traditional algorithms due to the inherent limitation of the model and to add a security component during the processing. Simulations are still a big part of membrane computing and they are useful to extract the right conclusions about the new model. In particular, this model is a great candidate to be applied to complex models that require an aggregated solution that is part of other subsolutions and super solutions as long as the defined rules in the MCA are followed. The encrypting module makes sure that the pro- cessed information is protected by applying the proposed encrypted method. The necessity of opening the line of research is out of question. The field is growing and new experiments are required. Encrypted MCA systems are provided as a natural solution to upgrade the nature of membrane computing by not only taking advantage of the properties of the membrane cells but also applying new cybersecurity principles. The future work will be involving complex problems in complex aggregated structures when working with encrypting algorithms. That could be a hit in applied sciences, especially when compared with traditional methods that are designed to speed computational processing such as supercomputing or parallel computing. There

seems to be close to reaching the limit when racing for processing, however, emerging models such as encrypted MCA can not only change the way of processing information but also provide the necessary security for a robust system.

References

1. Adleman LM (1994) Molecular computation of solutions to combinatorial problems. Science 266(5187):1021–1024. https://doi.org/10.1126/science.7973651
2. Albert AA, Blas NG, de Mingo López LF (2015) Natural combination to trade in the stock market. Soft Comput 20(6):2433–2450. https://doi.org/10.1007/s00500-015-1652-2
3. Albert AA, de Mingo López LF, Blas NG (2019) Multilinear weighted regression (mwe) with neural networks for trend prediction. App Soft Comput 82(105):555. https://doi.org/10.1016/j.asoc.2019.105555https://www.sciencedirect.com/science/article/pii/S1568494619303357
4. Albert AA, Díaz-Flores E, de Mingo López LF et al (2021) An in vivo proposal of cell computing inspired by membrane computing. Processes 9(3):511. https://doi.org/10.3390/pr9030511
5. Amir-Kroll H, Sadot A, Cohen IR, et al (2008) Gemcell: A generic platform for modeling multi-cellular biological systems. Theor Comput Sci 391(3):276–290. https://doi.org/10.1016/j.tcs.2007.11.014, converging Sciences: Informatics and Biology
6. Arteta A, Castellanos A, Martinez A (2010) Membrane computing: non deterministic technique to calculate extinguished multisets of objects. Int J Inf Technol Knowl 4(1):30–40
7. Arteta A, Mingo LF, Gómez N (2013) Membrane systems working with the p-factor: Best strategy to solve complex problems. Adv Sci Lett 19(5):1490–1494. https://doi.org/10.1166/asl.2013.4453
8. Arteta A, Mingo LF, Castellanos J (2018) An isomorphism based algorithm to solve complex problems
9. Arteta A, Mingo LF, Gomez N, et al (2019) Membrane computing aggregation (MCA): An upgraded framework for transition p-systems. In: Bio-inspired information and communication technologies. Springer International Publishing, pp 195–207. https://doi.org/10.1007/978-3-030-24202-2_15
10. Balázsi G, Barabási AL, Oltvai ZN (2005) Topological units of environmental signal processing in the transcriptional regulatory network of escherichia coli. Proc Natl Acad Sci 102(22):7841–7846. https://doi.org/10.1073/pnas.0500365102
11. Bashor CJ, Horwitz AA, Peisajovich SG et al (2010) Rewiring cells: Synthetic biology as a tool to interrogate the organizational principles of living systems. Annu Rev Biophys 39(1):515–537. https://doi.org/10.1146/annurev.biophys.050708.133652
12. Blow N (2009) Untangling the protein web. Nature 460(7253):415–417. https://doi.org/10.1038/460415a
13. Bolker JA (2005) Defining a meeting place: Modularity in development and evolution. Evolution 59(6):1383–1386. http://www.jstor.org/stable/3448914
14. Cao H, Romero-Campero FJ, Heeb S et al (2010) Evolving cell models for systems and synthetic biology. Syst Synth Biol 4(1):55–84. https://doi.org/10.1007/s11693-009-9050-7
15. Carroll SB (2005) Endless Forms Most Beautiful: The New Science of Evo Devo and the Making of the Animal Kingdom. W. W, Norton and Company
16. Danchin A (2009) Bacteria as computers making computers. FEMS Microbiol Rev 33(1):3–26. https://doi.org/10.1111/j.1574-6976.2008.00137.x
17. Federici D, Downing K (2006) Evolution and development of a multicellular organism: Scalability, resilience, and neutral complexification. Artif Life 12(3):381–409. https://doi.org/10.1162/artl.2006.12.3.381
18. de Frutos JA, Arroyo F, Arteta A (2009) Usefulness states in new p system communication architectures. Lecture Notes in Computer Science pp 169–186. https://doi.org/10.1007/978-3-540-95885-7_13
19. de Frutos JA, Fernández LS, Luengo C et al (2010) Improving active rules performance in new p system communication architectures. Int J Inf Technol Knowl 4(1):3–17
20. Han JDJ, Bertin N, Hao T et al (2004) Evidence for dynamically organized modularity in the yeast protein-protein interaction network. Nature 430(6995):88–93. https://doi.org/10.1038/nature02555
21. Haynes KA, Silver PA (2009) Eukaryotic systems broaden the scope of synthetic biology. J Cell Biol 187(5):589–596. https://doi.org/10.1083/jcb.200908138
22. Ho L, Crabtree GR (2010) Chromatin remodelling during development. Nature 463(7280):474–484. https://doi.org/10.1038/nature08911
23. Huh D, Matthews BD, Mammoto A et al (2010) Reconstituting organ-level lung functions on a chip. Science 328(5986):1662–1668. https://doi.org/10.1126/science.1188302
24. Hunter P, Robbins P, Noble D (2002) The IUPS human physiome project. Eur J Physiol 445(1):1–9. https://doi.org/10.1007/s00424-002-0890-1
25. Kondo S, Miura T (2010) Reaction-diffusion model as a framework for understanding biological pattern formation. Science 329(5999):1616–1620. https://doi.org/10.1126/science.1179047
26. Luscombe NM, Madan Babu M, Yu H et al (2004) Genomic analysis of regulatory network dynamics reveals large topological changes. Nature 431(7006):308–312. https://doi.org/10.1038/nature02782
27. Ma'ayan A, Blitzer RD, Iyengar R (2005) Toward predictive models of mammalian cells. Annu Rev Biophys Biomol Struct 34(1):319–349. https://doi.org/10.1146/annurev.biophys.34.040204.144415
28. Noble D (2002) Modeling the heart: From genes to cells to the whole organ. Science 295(5560):1678–1682
29. Oda K, Kimura T, Matsuoka Y et al (2004) Molecular interaction map of a macrophage. AfCS Research Reports 2(14):1–12
30. Palmer AR (2004) Symmetry breaking and the evolution of development. Science 306(5697):828–833. https://doi.org/10.1126/science.1103707
31. Păun G (2000) Computing with membranes. J Comput Syst Sci 61(1):108–143. https://doi.org/10.1006/jcss.1999.1693, https://www.sciencedirect.com/science/article/pii/S0022000099916938
32. Purnick PEM, Weiss R (2009) The second wave of synthetic biology: from modules to systems. Nat Rev Mol Cell Biol 10(6):410–422. https://doi.org/10.1038/nrm2698
33. Ray LB (2010) Evolving bacteria and computers. Science Signaling 3(123):ec160. https://doi.org/10.1126/scisignal.3123ec160
34. Sprinzak D, Lakhanpal A, LeBon L et al (2010) Cis-interactions between notch and delta generate mutually exclusive signalling states. Nature 465(7294):86–90. https://doi.org/10.1038/nature08959