An isomorphism based algorithm to solve complex problems

ALBERTO ARTETA
Computer Science Department
Troy University
University Avenue, Troy Alabama 36079
UNITED STATES
aarteta@troy.edu

LUIS F MINGO, JUAN CASTELLANOS
Information and Systems Department
Polytechnic University of Madrid
Crtra Valencia km 7 28031 Madrid,
SPAIN
lfmingo@eui.upm.es,jcastellanos@fi.upm.es

Abstract: - During years proper memory utilization has been the differential factor for algorithms that try to solve complex problems in optimal time. Computational complexity is currently measured in time and space. The way that algorithms are built can make a huge difference when obtaining desired solutions to problems. Memory engineering based algorithms reveal themselves as essential when fast results are needed and offer unlimited options to improve whatever bioinspired algorithm that based its performance in terms of time and space. This work focuses on creating a mathematical generic strategy by building structures through the use of an isomorphism that optimize the memory utilization and the resolution time of bioinspired models when dealing with high computational problems. Therefore offers a great help when solving complex and known issues.

Key-words— Bioinspired models, General Optimization , Complex problems resolution

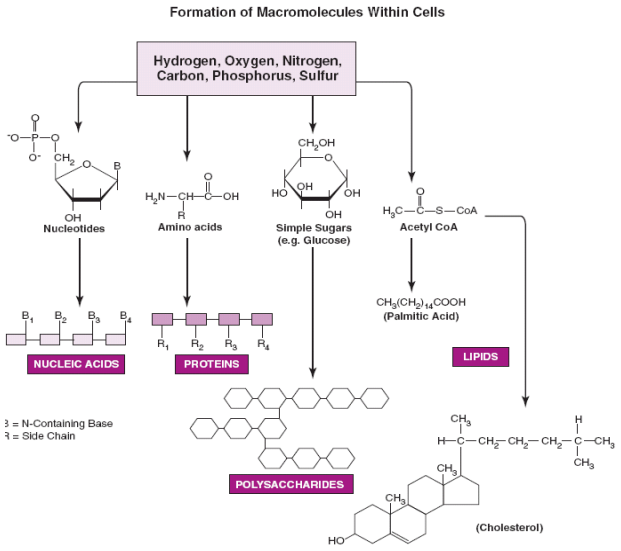
1 Introduction

The concept of transition p-system was first introduced in a 1998 by Gheorghe Paun, whose last name is the origin of the letter P in 'P Systems'. Variations on the P system model created what we call 'membrane computing.'

This model was inspired by biology, however P systems were used as a computational model, instead of a biological model. Nowadays there are some attempts of creating new biological models through the use of p-systems. In [1] there are some keywords related to membrane computing such as parallelism, distributed applications, algorithms. Paun avoids stating that there are implementations of p-systems. This happens because the model he proposes has an inherent parallelism and non determinism. Those characteristics cannot be implemented in a conventional computer. [1].

Regarding simulators, [2] remarks: simulations on conventional computers can be useful too: In fact all these programs were used on biological applications and they can also play an important role for didactic purposes and to establish new researching areas.

The living cells we see that they process food and nutrients in a very characteristic way. Living cells get atoms or molecules and then react by producing other atoms and molecules.



This process has been studied and it has also been tried to implement into computers. Living cells act and process nutrients according to certain rules that take place in a parallel and in a non deterministic manner. This model has been copied by many researchers to establish new machines that process information in the same way. In 1998 George Paun studied the described biological model and then was able to create what we call nowadays 'Membrane computing'. This was created mainly inspired on the processes that take place inside of the living cells. After analyzing such processes, it is noticeable that they have properties that can be used into a computational model. This model was a revolution because it opened new researching areas for solving NP-complete problems.

Foundations and organizations were aware of this new revolutionary way of computing and encouraged the research on that direction. Examples of these are EMCC (European Molecular Computing Consortium) and The Consortium for Biomolecular Computing.

In 2004 the official p-system web http://psystems.disco.unimib.it was created; since then, it has been supported by the EMCC

In February 2003 membrane computing was defined as "the most revolutionary way of computing within ICT". This was said by the Institute of Scientific Information.

The transition P-systems have been suffered a transformation themselves. This transformation comes from the desire of collecting more biological properties. Moreover the use of automats has been added to improve the performance of p-systems. One of the main goals for the p-systems to achieve is the resolution of Np completes problems in polynomial time.

Natural computing group in Madrid is currently working on implementing p-systems in specialized hardware. The parallelism degree achieved is high and the temporary results obtained are promising. Also, the idea of reducing to practice the p-systems is more realistic.

Algorithms for applying evolution rules are getting improved too. A new algorithm that obtains optimal results when simulating the rules application phase, is about to be published. This algorithm use combinatorial laws and statistics.

Np complete problems have been solved in polynomial time and sometimes even in linear. We are working closer with biological areas and now there is a membrane computing application in biology. It is believed that membrane computing can be a reliable framework to model living cells processing.
Moreover, this group has avoided approaching p-systems as an isolated technology. has worked on scenarios in where the p-systems can work together with multiagent systems. There is another paper to be published in where p-systems works with autonomous robots to find optimal solutions to known problems

2 Definitions

Patterns

$$N \otimes N \rightarrow A \qquad A \subseteq \mathrm{P}(\mathrm{N})$$
$$[i, j] = \{ k \in N \mid i \leq k \leq j \quad i, j \in N \}$$

Set of patterns $S \subseteq \mathrm{P}(\mathrm{N})$ is a *Set of patterns is defined as the set:*

$$\{[a_i, b_j] \mid \exists n, m \in \mathrm{N}, i \leq n \; j \leq m \; a_i, b_j, i, j \in \mathrm{N}\}$$

Set of set of patterns

$$SS = \{S_i \; i \in \mathrm{N} \; S_i \; is \; a \; set \; of \; patterns \mid \exists n \in \mathrm{N} \; i \leq n\}$$

.

Observation
Given a region R and alphabet of objects U, and R (U, T) set of evolution rules over U and targets in T re`presented as follows:

$$r_1 : a_1^{u_{11}} a_2^{u_{12}} . a_n^{u_{1n}} \rightarrow C_1$$
$$r_2 : a_1^{u_{21}} a_2^{u_{22}} . a_n^{u_{2n}} \rightarrow C_2$$
$$\ldots \qquad \rightarrow \ldots.$$
$$r_m : a_1^{u_{m1}} a_2^{u_{m2}} . a_n^{u_{mn}} \rightarrow C_m$$

There is always a set of set of patterns $SS_{R(U,T)}$ associated to it. This set of set of patterns contains all the possible extinguished multisets and it is obtained by expanding the formula included in the definition of extinguished multiset:

$$\bigcap_{l=1}^{m}\left[\bigcup_{i=1}^{n}\left(u_i - \sum_{j=1}^{m}(k_j \cdot u_{ji}) \le u_{li}\right)\right]$$

$$\begin{pmatrix}
[[0,u_{11}],[0,u_{12}]...[0,u_{1n}]],[[0,u_{11}],[0,u_{12}]...[0,u_{2n}]],...,[[0,u_{11}],[0,u_{12}]...[0,u_{mn}]],\\
...\\
[[0,u_{11}],[0,u_{22}]...[0,u_{1n}]],[[0,u_{11}],[0,u_{22}]...[0,u_{2n}]],...,[[0,u_{11}],[0,u_{22}]...[0,u_{mn}]],\\
...\\
[[0,u_{11}],[0,u_{m2}]...[0,u_{1n}]],[[0,u_{11}],[0,u_{m2}]...[0,u_{2n}]],...,[[0,u_{11}],[0,u_{n2}]...[0,u_{mn}]],\\
...\\
[[0,u_{21}],[0,u_{12}]...[0,u_{1n}]],[[0,u_{21}],[0,u_{12}]...[0,u_{2n}]],...,[[0,u_{21}],[0,u_{12}]...[0,u_{mn}]],\\
...\\
[[0,u_{21}],[0,u_{22}]...[0,u_{1n}]],[[0,u_{21}],[0,u_{22}]...[0,u_{2n}]],...,[[0,u_{21}],[0,u_{22}]...[0,u_{mn}]],\\
...\\
[[0,u_{21}],[0,u_{m2}]...[0,u_{1n}]],[[0,u_{21}],[0,u_{m2}]...[0,u_{2n}]],...,[[0,u_{21}],[0,u_{n2}]...[0,u_{mn}]],\\
...\\
[[0,u_{m1}],[0,u_{12}]...[0,u_{1n}]],[[0,u_{1n}],[0,u_{12}]...[0,u_{2n}]],...,[[0,u_{m1}],[0,u_{21}]...[0,u_{mn}]],\\
...\\
[[0,u_{m1}],[0,u_{22}]...[0,u_{1n}]],[[0,u_{1n}],[0,u_{22}]...[0,u_{2n}]],...,[[0,u_{m1}],[0,u_{22}]...[0,u_{mn}]],\\
...\\
[[0,u_{m1}],[0,u_{m2}]...[0,u_{1n}]],[[0,u_{1n}],[0,u_{m2}]...[0,u_{2n}]],...,[[0,u_{m1}],[0,u_{2n}]...[0,u_{mn}]],
\end{pmatrix}$$

LinLinear Multisets isomorphism Φ_1

Building Φ_1, as a part of the physical evolution rule isomorphism is created as a function which has an m-dimensional set of natural numbers as input and an n-dimensional set of natural numbers as output, where m is the number of evolution rules and n is the number of symbols included in a given multiset of objects.

Definition: Linear Multisets function φ_1

Let U= $\{a_i \mid i = 1,..n\}$ be a set of objects. Let T beset of targets. Let $\omega = a_1 a_{2..} a_n$ be a multiset of objects and let x_i be the multiplicity of a_i. Let R (U, T) be a multiset of evolution rules with objects in U and targets in T. Let m be the number of evolution rules within R (U, T). Let k_i be the number of times that the rule $r_i \in R(U,T)$ is applied. Then φ_1 is defined as:

$$\begin{array}{lll}\varphi_1 : N^m & \rightarrow & N^n \\ (k_1, k_2,.., k_m) & \rightarrow & (x_1, x_2,.., x_n)\end{array}$$

$$\varphi_1(k_1,k_2,..k_m,) \equiv \begin{pmatrix} u_{11} & u_{21} & ..u_{m1} \\ u_{12} & u_{22} & ..u_{m2} \\ \dots & \dots & \dots \\ u_{1n} & u_{2n} & ..u_{mn} \end{pmatrix}\begin{pmatrix} k_1 \\ k_2 \\ \dots \\ k_m \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}$$

Once the function φ_1 is created, the isomorphism Φ_1 is defined as follows:

Definition: Linear Multisets isomorphism Φ_1

Let φ_1 be the multiset linear function related to a given evolution rules set $\Phi_1 : N^n \rightarrow N^n$

$$\Phi_1(x) = \begin{Bmatrix} \varphi_1(x) & \neg\exists y \neq x \mid \varphi_1(y) = \varphi_1(x) \\ random\{\varphi_1(x), \varphi_1(y)\} & \exists y \neq x \mid \varphi_1(y) = \varphi_1(x) \end{Bmatrix}$$

Definition

Physical evolution rules linear Isomorphism is then defined as follows:

$$\begin{array}{lll}\Phi = \Phi_1 \circ \Phi_2 : N^m & \rightarrow & P(N^n) \\ (k_1, k_{2,}..., k_m) \xrightarrow{\Phi_1} (x_1, x_2,.., x_n) \xrightarrow{\Phi_2} (x_1, x_2,.., x_n) - SS_{R(U,T)} & & \end{array}$$

Based on the previous definitions, the following one is established and created to take full advantage of the virtual memory.

Definiion: Virtual linear Multisets function

$$\Phi_{1vir} : N^n \rightarrow P(N^m)$$

$$[x_1, x_2,.., x_n] \rightarrow \begin{bmatrix} k_1, k_2,.., k_m \\ k_1', k_2',.., k_m' \\ \dots \\ k_1'^p, k_2'^p,.., k_m'^p \end{bmatrix}$$

Given an input $x = (x_1, x_2,.., x_n) \in N^n$, it returns the set of numbers $k = (k_1, k_2,.., k_m) \in N^m / \varphi_1(k) = x$

Definition: Virtual linear Pattern function

$$\Phi_{2virt}: P(N^m) \rightarrow N^n$$
$$\begin{pmatrix} [[0,u_{11}],...[0,u_{m1}]],..,[[0,u_{11}],..[0,u_{mn}]], \\ ... \\ [[0,u_{12}],..[0,u_{m1}]],..,[[0,u_{12}],..[0,u_{mn}]], \end{pmatrix} \rightarrow \begin{bmatrix} x_1, x_2,.., x_n \\ x_1', x_2',.., x_n' \\ ... \\ x_1'^p, x_2'^p,.., x_n'^p \end{bmatrix}$$

Given a *set of set of patterns* as the input it returns a set of numbers $x = (x_1, x_2,.., x_n) \in N^n$. The elements of this resulting set are all the combinations of all the possible

$x^j = (x_1, x_2,.., x_n) \in N^n$ where $x_i^j \in pattern\ (i) \in SP\ (j)$ of a set of patterns contained in the matrix of set of patterns.Now it is possible to build the physical and virtual linear structures from the multisets isomorphism.

3. Isomorphism based structure

The structure is created as follows.

$$L[\Phi(k_1, k_2,.., k_m)] = \begin{cases} (k_1,k_2,...,k_m) & L[\Phi(k_1,k_2,...,k_m)] = \phi \\ random\{(k_1,k_2,...,k_m), no\,action\} & L[\Phi(k_1,k_2,...,k_m)] \neq \phi \end{cases}$$

When L (i, j) already has a value, then, a random election must be done. This election can be either overwriting the old value with the new one, or to leave the old value (No action).This linear structure has to comply with having all possible numbers, up to a combination of benchmarks, reasonably high. Each symbol $\{X_i \mid i = 1,.. n\}$ will have a benchmark. The combination of all the benchmarks will define the number of entries that the linear structure has. Each entry stores the values $\{k_1,.. k_m\}$. These values indicate the number of times that an evolution rule should be applied to an initial multiset in order to obtain an extinguished multiset.

The linear structure L_Φ must guarantee that it contains all the entries corresponding to the combination number of all benchmarks associated to the symbols $\{X_i \mid i = 1,.. n\}$, i.e. it can not have non functioning entries. That could damage response timing and it would increase the computational complexity in terms of time. It's necessary to prove that this structure has not null values.

Let L_Φ be a linear structure built from the *evolution rules isomorphism* and a certain *V* multiset of objects and *R (U, T)* multiset of evolution rules. $\rightleftarrows$

$$L_\Phi[X_1, X_2,.., X_n] \in N^m \ \forall (X_1, X_2,.., X_n) \in N^n \ X_i \leq X_i'$$
$$\vdash \forall i \leq n \ \ benchmark\ (X_i) = X_i' \ \ |R(U,T)| = m \ and \ n = |V|$$

Proof

Let $\varphi_1 \equiv \begin{pmatrix} A_{11} & .. A_{m1} \\ .. & .. \\ A_{1n} & .. A_{mn} \end{pmatrix}$ be the *function* associated to *R (U, T)*

$$\begin{pmatrix} A_{11} & .. A_{m1} \\ .. & .. \\ A_{1n} & .. A_{mn} \end{pmatrix} \begin{pmatrix} k_1 \\ .. \\ k_m \end{pmatrix} \Rightarrow$$

$$\begin{matrix} A_{11} k_1 + & + .. A_{m1} k_m & (1) \\ .. & ... & (i) \\ A_{1n} k_1 + & + .. A_{mn} k_m & (n) \end{matrix}.$$

A resulting set of n-sums is obtained. Let A_i be = {min A_{ij} of the sums (i)} .Moreover, let $P(A_{ij})$ be the Matrix of set of patterns resulting from *R (U, T).* $\forall i \in N\ i \leq n, \exists S_{R(U,T)}\ set\ of\ patterns / S_{R(U,T)}[i] = [0, A_i]$. Proof is trivial

Thus,
$\forall A_{ij} \neq A_i\ A_{ij} = 0 \Rightarrow \exists j \in N\ j \leq m / A_i \cdot k_j + [0, A_i] = x\ \forall x \in N$.

Observation:

This proves that when following this method, any natural number from $\{k_1,.., k_m\} \in N^m$ can be generated. Thus, the structure does not have null

values from any entry. As the process is an isomorphism, any natural number between 0 and a given benchmark will be uniquely related to a combination of $\{k_1,.., k_m\} \in N^m$ where *m* is the number of evolution rules included in *R (U, T).* Moreover the calculation of *extinguished multisets* will be immediate. When $\{k_1,.., k_m\} \in N^m$ are found, applying the evolution rule r_i a number of k_i times to the initial multiset, will calculate them.

Given $V= \{\{X_i \mid i = 1,.. n\}$ be a *multiset of symbols and given R (U, T),* a multiset of evolution rules. *Given the set* $\{k_i \in N$ *the number of times that the evolution rule* r_i *is applied over the initial multiset}* the following evolution rules function is defined:

$$\Phi_{virt} = (\Phi_{1\,virt} \circ \Phi_{2\,virt}) (P_{R(U,T)}(A_{ij})$$

Consistency theorem

The linear structure L_Φ must guarantee that it contains all the entries corresponding to the combination number of all benchmarks associated to the symbols $\{X_i \mid i = 1,..n\}$, i.e. it can not have non functioning entries. That could damage response timing and it would increase the computational complexity in terms of time. It's necessary to prove that this structure has not null values.

Let L_Φ be a linear structure built from the *evolution rules isomorphism* and a certain *V* multiset of objects and *R (U, T)* multiset of evolution rules. $\rightrightarrows$

$$L_\Phi[X_1, X_2,.., X_n] \in N^m \ \forall (X_1, X_2,.., X_n) \in N^n \ X_i \le X_i^{'}$$
$$\forall i \le n \ \ benchmark(X_i) = X_i^{'} \quad |R(U,T)| = m \ and \ n = |V|$$

Proof

$$\varphi_1 \equiv \begin{pmatrix} A_{11} & ..A_{m1} \\ .. & .. \\ A_{1n} & ..A_{mn} \end{pmatrix}$$

Let be the *function* associated to *R (U, T)*

$$\begin{pmatrix} A_{11} & ..A_{m1} \\ .. & .. \\ A_{1n} & ..A_{mn} \end{pmatrix} \begin{pmatrix} k_1 \\ .. \\ k_m \end{pmatrix} \Rightarrow$$

$$\begin{array}{lll} A_{11}k_1 + & + .. A_{m1}k_m & (1) \\ .. & ... & (i) \\ A_{1n}k_1 + & + .. A_{mn}k_m & (n) \end{array}.$$

A resulting set of n-sums is obtained. Let A_i be = {min A_{ij} of the sums (i)} .Moreover, let $P(A_{ij})$ be the Matrix of set of patterns resulting from *R (U, T).*

$\forall i \in N \ i \le n, \exists S_{R(U,T)}$ *set of patterns* $/ S_{R(U,T)}[i] = [0, A_i]$. Proof is trivial

Thus,

$$\forall A_{ij} \ne A_i \ A_{ij} = 0 \Rightarrow \exists j \in N \ j \le m / A_i \cdot k_j + [0, A_i] = x \ \forall x \in N$$

.

Remark:

This proves that when following this method, any natural number from $\{k_1,..k_m\} \in N^m$ can be generated. Thus, the structure does not have null values from any entry. As the process is an isomorphism, any natural number between 0 and a given benchmark will be uniquely related to a combination of $\{k_1,..k_m\} \in N^m$ where *m* is the number of evolution rules included in *R (U, T).* Moreover the calculation of *extinguished multisets* will be immediate. When $\{k_1,..k_m\} \in N^m$ are found, applying the evolution rule r_i a number of k_i times to the initial multiset, will calculate them.

corresponding number in square brackets as shown at the end of this sentence [1].

4 Virtual structure

Given $V=\{\{X_i \mid i=1,..n\}$ be a *multiset of symbols and given R (U, T),* a multiset of evolution rules. *Given the set* $\{k_i \in N$ *the number of times that the evolution rule* r_i *is applied over the initial multiset}* the following evolution rules function is defined:

$$\Phi_{virt} = (\Phi_{1\,virt} \circ \Phi_{2\,virt})(P_{R(U,T)}(A_{ij}))$$

.

Once the virtual function has been constructed, the new virtual linear structure must be built this way:

$$L_{virt}\left[\Phi_{2virt}\left(P_{R(U,T)}(A_{ij})\right)\right] = \Phi_{1virt} \circ \Phi_{2virt}\left(P_{R(U,T)}(A_{ij})\right) append\left[L_{virt}\left[\Phi_{2virt}(M)\right]\right]$$

=

When $L_{\Phi\,virt}$ already has a value, then the new values are included and are appended to the existing ones. This means that in each entry of the virtual linear structure, there will be different values

$$(k_1, k_2, .., k_m)\ append\ [(k_1', k_2', .., k_m')]\ append\ ...\ append\ [(k_1'^p, k_2'^p, .., k_m'^p)]$$

A concatenation of $(k_1, k_2, .., k_m)$ values is included in each cell of the virtual linear structure. This linear structure has to comply with having all possible numbers up to a combination of benchmarks reasonably high. Each symbol $\{X_i \mid i=1,..n\}$ has a benchmark. The combination of all the benchmarks defines the number of entries that the linear structure has. Each entry stores then a concatenation of values $\{k_1, .., k_m\}$. These values indicate the number of times that an evolution rule should be applied to an initial multiset in order to obtain an extinguished multiset. After proving the consistency of the physical structure proving the consistency of the virtual one is trivial.

5 Example

This example describes the steps to create both *structures*. In the first part *isomorphism* is defined, it will then creates the structure and place it in the RAM.

In the second part, another function is built and then allocated within the *virtual memory*. The example considers a situation where the number of symbols is less than the number of evolution rules, which is the most interesting case for this analysis.

$V = \{X_1, X_2\}.\quad R(U,T) = \{r_1, r_2\}$

$r_1 : X_1^{\,4} X_2^{\,3} \rightarrow C_1$

$r_2 : X_1^{\,3} X_2^{\,2} \rightarrow C_2$

$r_3 : X_1^{\,1} X_2^{\,1} \rightarrow C_3$

From this, the set of patterns obtained is as follows:

$$S_{R(U,T)} = \{[[0],[0..\infty)],[[0..\infty),[0]]\}$$

$$\varphi_1(k_1,k_2,k_3) \equiv \begin{pmatrix} 4 & 3 & 1 \\ 3 & 2 & 1 \end{pmatrix} \begin{pmatrix} k_1 \\ k_2 \\ k_3 \end{pmatrix}$$

Let's set (X_1, X_2) benchmarks as (10, 12)

Let (k_1, k_2, k_3) be selected randomly. For example:

$(k_1, k_2, k_3) = (0, 0, 1)$

$\Phi_1(k_1,k_2,k_3) = (X_1, X_2)$

$\Phi_1(0,0,1) = (1,1); \Phi_1(0,1,0) = (3,2); \Phi_1(1,0,0) = (4,3); \Phi_1(0,1,1) = (4,3);$

$\Phi_1(1,1,0) = (7,5); \Phi_1(1,0,1) = (5,4); \Phi_1(1,1,1) = (8,6); \Phi_1(2,0,0) = (8,6);$

$\Phi_1(0,0,2) = (2,2); \Phi_1(0,2,0) = (6,4); \Phi_1(2,0,1) = (9,7); \Phi_1(2,0,2) = (10,8)$

$\Phi_1(0,1,2) = (5,4); \Phi_1(0,2,1) = (7,5); \Phi_1(2,1,1) = (12,9); \Phi_1(2,1,2) = (12,9);$

$\Phi_1(2,2,1) = (15,11); \Phi_1(2,2,0) = (14,10); \Phi_1(1,2,0) = (10,7);$

$\Phi_1(1,2,1) = (11,8); \Phi_1(1,2,2) = (12,9); \Phi_1(2,2,1) = (15,11);$

$\Phi_1(2,1,0) = (11,8); \Phi_1(2,2,2) = (16,12)$.....

Once Φ_1 is completed, the next step is building the function Φ_2

$$\Phi_2(X_1, X_2) = (X_1, X_2) - P(X_1, X_2)$$
$$\Phi_2(1,1) = \{[[0..1],1],[1,[0..1]]\};$$
$$\Phi_2(2,2) = \{[[0..2],2],[2,[0..2]]\};$$
$$\Phi_2(3,2) = \{[[0..3],2],[3,[0..2]]\};$$
$$\Phi_2(4,3) = \{[[0..4],3],[4,[0..3]]\}$$
$$\Phi_2(5,4) = \{[[0..5],4],[5,[0..4]]\}$$
$$\Phi_2(6,4) = \{[[0..6],4],[6,[0..4]]\}... \quad \Phi_2 =$$

Now there is enough information to build the physical linear structure L_Φ in this example. A way to build the structure could be:

$$= L_\Phi(x) =$$
$$L_\Phi(0,1)=001, L_\Phi(1,0)=001, L_\Phi(1,1)=001, L_\Phi(2,0)=002,$$
$$L_\Phi(2,1)=010, L_\Phi(2,2)=002, L_\Phi(3,2)=010, L_\Phi(3,3)=010,$$
$$L_\Phi(4,3)=100, L_\Phi(5,4)=101...$$

Once this structure is created, it will be allocated in the RAM. At this moment it is possible to build the virtual linear structure. The process is similar to the one above. The main difference will be that in each cell a concatenation of values will be stored instead of just one value.

The structure is built as follows:

$$L_{\Phi virt}(x) =$$
$$L_{\Phi virt}(0,1) = 0,0,1\ L_{\Phi virt}(1,0) = 0,0,1\ L_{\Phi virt}(1,1) = 0,0,1\ L_{\Phi virt}(2,0) = 0,0,2$$
$$L_{\Phi virt}(2,1) = 0,1,0\ L_{\Phi virt}(2,2) = 0,1,0 \quad 0,0,2\ L_{\Phi virt}(3,2) = 0,1,0\ L_{\Phi virt}(3,3) = 0,1,0$$
$$L_{\Phi virt}(4,3) = 1,0,0 \quad 0,1,1 L_{\Phi virt}(5,4) = 1,0,1 \quad 0,1,2 \quad 0,2,0....$$

6 Technique Analysis

Building the structures guarantees, that any input value $\{X_1,..,X_n\} \in N^n$ has a direct relation with $\{k_1,..,k_m\} \in N^m$. This implies that the structure can be built after certain evolution rules are given. In this way, any algorithm that intends to calculate an extinguished multiset from an initial multiset of objects can generate the linear structure L_Φ during the compilation time.

During the compiling stage of an algorithm, two new structures are created. When the multiplicities of the initial multisets are provided as input values $\{X_1,..,X_n\} \in N^n$ of a given algorithm, this algorithm will search the entry $[X_1,..,X_n]$ of the physical structure. Thus the algorithm returns $L_\Phi[X_1,..,X_n]$. After the value is returned, a new value is generated by searching the entry $L_{\Phi virt}[X_1,..,X_n]$ and then selecting randomly one of the possible combinations of $\{k_1,.., k_m\} \in N^m$ stored in that entry. Once it is selected, this value would overwrite the value [illegible]. The way that the structure has been created, assures that for each input values, different outputs might be generated. Thus, the use of the linear structure preserves the non-determinism, which a characteristic of the Paun biological model.

In both structures, the number of cells that L_Φ requires is: $\prod_{i=1}^{n} benchmark\ (X_i)$ where n is the number of symbols of the current multiset. The combination of the number of symbols and the selected benchmarks of each symbol will determine the number of cells that the linear structure must have. Thus, the combination of these two factors has a major influence in the amount of memory used by L_Φ.

Within the last years, algorithms have been developed to calculate extinguished multisets. They have achieved a computational complexity equal to $\Theta(m)$ where m is the number of evolution rules involved.

When the number of evolution rules is high, the execution time can take too long. On the contrary, the best scenario for this technique is when the combination between number of symbols and their benchmarks is low, regardless the number of evolution rules and the maximum applicability benchmark of any evolution rule.

For instance, let us consider a hypothetical scenario where the multiset of symbols= $S=\{X_1,X_2\}$, and the number of evolution rules is 10^{12}. Let us also consider that the maximum value that an evolution rule can be applied is 10^{24}.

Under these conditions, an algorithm as the *fast linear algorithm* [16] is inefficient as its complexity is O (m) where m is the number of evolution rules. On the contrary, an algorithm that uses this technique can find extinguished multisets with a constant complexity order. When using this technique, it is necessary to reserve 124 bits for each structure's cell as per: $10^{12} \cdot 10^{24} = 10^{36} \leq 2^{4^{36}} = 2^{124}$.

This means 15 bytes per cell .Let us set a benchmark of 30000 for each symbol X_i of the multiset. Considering that there are two symbols within our current Multiset: $S=\{X_1,X_2\}$, then: $30000 \cdot 30000 = 9 \cdot 10^8$ number of cells of L_Φ. Thus, the amount of memory needed to create L_Φ is

$$9 \cdot 10^8 \ cells \cdot 15 \ (bytes / cell) = 135 \cdot 10^8 \ bytes = 13.5 Gb$$

Thus, when having 13.5 Gb any algorithm using this technique can find extinguished multisets with complexity order O(1) as long as Input values for X_1 *and* X_2 multiplicities are <30000. This will occur regardless the number of evolution rules.

The amount of memory needed to create the *virtual structure* increases as every cell can store more than one set of numbers, i.e. $\{k_1, k_2.., k_m \quad k_1', k_2'.., k_m' \quad k_1'',...\}$, Let us assume that the maximum number of combinations $k_1,k_2..,k_m$ stored in each entry is 10^{48}. This means $10^{50} \cdot 10^{12} \cdot 10^{24} = 10^{86} \leq 2^{4\,86} = 2^{344}$. This means 43 bytes per cell.

$$L_{\Phi virt} =$$

$$9 \cdot 10^8 \ cells \cdot 43 (bytes / cell) = 387 \cdot 10^8 \ bytes = 38.7 Gb$$

Once this is done, both structures are built. The physical structure relates the number of times that each evolution rule should be applied to the multiplicity input values of the initial multiset. This occurs as long as these values are always either less than or equal to their benchmark.

An algorithm, as the one based on the maximum applicability benchmark, could create the structures during compilation stage. If the number of evolution rules is high, it would be worth considering creating the structures. This finds the corresponding values to the number of times that each rule should be applied in order to obtain an extinguished multiset. This happens as long as the input values of the object's multiplicities are lower than the previously fixed benchmarks.

7 Algorithm

Following is the code that returns the multisets as outputs

(1) $X, Y \leftarrow Multiplicity(R(U,T))$
(2) *BEGIN*
(3) $output\left(L_\Phi(X,Y)\right)$
(4) *END*
(5) $L_\Phi(X,Y) \leftarrow L_{\Phi virt}(X,Y)$

The algorithm search in the physical structure the position(X, Y) which are the input values corresponding to the multiplicities of the initial multiset. When the value is returned, the algorithm finishes and a new value coming from the virtual structure overwrite the value stored in the position (X, Y), keeping the non deterministic nature of the system.

8 Conclusions and further work

Although it is clear that proper memory utilization is a great help to speed up the information processing, it is still necessary to build the right structures to optimize the performance; and that is not always easy. This paper contributes with a general technique that consists of building an isomorphism based structure that improves the performance of traditional algorithms when dealing with complex problems. The isomorphism matches the initial data sets multiplicities with the number of times that each rule should be applied in order to obtain solutions in optimal time; In that way, this method clearly reduces the execution times of the algorithms when finding maximal or extinguished multi data sets.

The nature of the isomorphism ensures that the main bioinspired system features are preserved and the overall system's functionality is not modified, the only difference between the first model and the second one is the performance. Although the idea of using memory resources to increase performance when solving complex problems is not new, defining the

right structure based on isomorphism can offer a promising way to generalize and to create a standard methods for optimizing the memory resources of traditional algorithms. Improving the design and the implementation of new isomorphisms reveal themselves as keys factors in the strategy to deal with complex problems by utilizing memory engineering.

References:

[1] Nidhal kamel taha el-omari "Scanned document image segmentation using back-propagation artificial neural network based technique" North Atlantic University Union (NAUN) International journal of computers and communications issue 3, volume 6, 2012

[2] Enhancing Parallel Recursive Brute Force Algorithm for Motif Finding , 103 Marwa Radad recent advances in computer science proceedings of the 6th wseas world congress: applied computing conference (acc '13) 103-11

[3] Dimitar ivanov, univis – "A 3d software system visualization using natural metaphors" pages: 107-119 International Journal of computers Issn:1998-4308 volume 8, 2014 North Atlantic University Union (NAUN)

[4] Gh. Păun, "Computing with Membranes", Journal of Computer and System Sciences, 61(2000), and Turku Center of Computer Science-TUCS Report nº 208, 1998.

[5] Gh. Păun, "Membrane computing. Basic ideas results, applications", Pre-Proceedings of First International Workshop on Theory and Application of P Systems, Timisoara (Romania), pp. 1-8, September , 2005.

[6] Kenneth Price, Rainer M. Storn, and Jouni A. Lampinen. Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series). Springer- Verlag New York, Inc., 2005.ISBN 3540209506..

[7] K.V. Price. Differential evolution: a fast and simple numerical optimizer. In Fuzzy Information Processing Society, 1996. NAFIPS. 1996 Biennial Conference of the North American, pages 524–527, 1996. doi: {10.1109/NAFIPS.1996.534790}.

[8] Lingian Pan, Carlos Martin-Vide "Solving multidimensional 0-1 knapsack problem by P systems with input and active membranes", Journal of Parallel and Distributed Computing Volume 65 , Issue 12 (December 2005)

[9] Lin, Shen, Kernigham BW An Effective Heuristic Algorithm for the Traveling-Salesman Problem". Operations Research 21 (2): 498–516. doi:10.1287/opre.21.2.498.Volume 21, Issue 2, March- April 1973

[10] A. Arteta, L.Fernandez, J.Gil "Algorithm for Application of Evolution Rules based on linear diophantine equations" Synasc 2008.Timisoara Romania September 2008

[11] Arroyo, A. Arteta,, A. Goñi. Calculating maximal multisets using RAM as support, Artificial life and Robotics 2010, Beppu Japan

[12] Alberto Arteta, Nuria Gomez Luis Fernando Mingo. ,Solving complex problems with a bioinspired model. Engineering Applications of Artificial Intelligence,Volume 24, Issue 6, September 2011, Pages 919–927

[13] Alberto Arteta , Angel Castellanos, Ana Martinez: Membrane computing: non deterministic technique to calculate extinguished multisets of objects. International Journal " Information Technologie s and Knowledge", Vol. 4, Number 1, 2010

[14] Chiang, Jui-Hao. Optimization Techniques for Memory Virtualization-based Resource Management Publisher: The Graduate School, Stony Brook University: Stony Brook, NY. Date: 1-Dec-12

[15] Ganon, Jalby Strategies for cache and local memory management by global

program transformation Journal of Parallel and Distributed Computing, Volume 5, Issue 5, October 1988, Pages 587–616

[16] L. Fernandez, J.Castellanos, F. Arroyo, J. tejedor, I Garcıa "New algorithm for application of evolution rules", Proceedings of the 2006 International Conference on Bioinformatics and Computational Biology, BIOCOMP'06, Las Vegas, Nevada, USA, 2006.