MASSIVE PARALLEL NETWORKS OF EVOLUTIONARY PROCESSORS AS NP-PROBLEM SOLVERS

Nuria Gómez Blas, Luis F. de Mingo and Eugenio Santos
Dept. Organizació y Estructura de la Información, Escuela Universitaria de Informática
Universidad Politécnica de Madrid, Crta. de Valencia km. 7, 28031 Madrid, Spain

Keywords: Natural Computing, Networks of Evolutionary Processors, Symbolic Systems.

Abstract: This paper presents a new connectionist model that might be used to solve NP-problems. Most well known numeric models are Neural Networks that are able to approximate any function or classify any pattern set provided numeric information is injected into the net. Concerning symbolic information new research area has been developed, inspired by George Paun, called Membrane Systems. A step forward, in a similar Neural Network architecture, was done to obtain Networks of Evolutionary Processors (NEP). A NEP is a set of processors connected by a graph, each processor only deals with symbolic information using rules. In short, objects in processors can evolve and pass through processors until a stable configuration is reach. Despite their simplicity, we show how the latter networks might be used for solving an NP-complete problem, namely the 3-colorability problem, in linear time and linear resources (nodes, symbols, rules).

1 INTRODUCTION

Natural sciences, and especially biology, represents a rich source of modeling paradigms. Well-defined areas of artificial intelligence (genetic algorithms, neural networks), mathematics, and theoretical computer science (L systems, DNA computing) are massively influenced by the behavior of various biological entities and phenomena. In the last decades or so, new emerging fields of so-called “natural computing” (Zandron, 2002; Paun, 2002; Ciobanu et al., 2005) identify new (unconventional) computational paradigms in different forms. There are attempts to define and investigate new mathematical or theoretical models inspired by nature (Robinson, 1992), as well as investigations into defining programming paradigms that implement computational approaches suggested by biochemical phenomena. Especially since Adleman’s experiment (Adleman, 1994), these investigations received a new perspective. One hopes that global system-level behavior may be translated into interactions of a myriad of components with simple behavior and limited computing and communication capabilities that are able to express and solve, via various optimizations, complex problems otherwise hard to approach.

The origin of networks of evolutionary processors is a basic architecture for parallel and distributed symbolic processing, related to the Connection Machine (Hillis, 1985) as well as the Logic Flow paradigm (Errico and Jesshope, 1994), which consists of several processors, each of them being placed in a node of a virtual complete graph, which are able to handle data associated with the respective node. All the nodes send simultaneously their data and the receiving nodes handle also simultaneously all the arriving messages, according to some strategies, see, e.g., (Fahlman et al., 1983; Hillis, 1985).

2 MASSIVE PARALLEL NEPS

Let V an alphabet over a set of symbols. A string x of length m belonging to an alphabet V is the sequence of symbols \( a_1 a_2 \cdots a_m \) where the symbol \( a_i \in V \) for all \( 1 \leq i \leq m \). The set of all strings that belong to V is denoted by \( V^* \) and the empty string is denoted by \( \epsilon \).

A network of evolutionary processors (Castellanos et al., 2001; Castellanos et al., 2003) of size n (NEP for short) is a construct \( \Sigma = \{ V, N_0, N_1, \cdots, N_n \} \), where V is an alphabet and processors \( N_i \) are connected within a graph.

A processor is defined by \( N_i = \{ M_i, A_i, PI_i, PO_i \} \) is the i-th evolutionary processor of the network. The parameters of every processor are:
- $M_i$ is a finite set of evolution rules of one of the following forms only (basic rules):
  - $a \rightarrow b, a, b \in V$ (substitution rules),
  - $a \rightarrow \varepsilon, a \in V$ (deletion rules),
  - $\varepsilon \rightarrow a, a \in V$ (insertion rules).

More clearly, the set of evolution rules of any processor contains either substitution or deletion or insertion rules. Context information can be added to $M_i$ in the following way (directional context rules):

- $ac \rightarrow bc, a, b \in V, c \in V^*$ (left substitution rules),
- $ca \rightarrow cb, a, b \in V, c \in V^*$ (right substitution rules),
- $ab \rightarrow a, b \in V, a \in V^*$ (left deletion rules),
- $ba \rightarrow ab, b \in V, a \in V^*$ (right insertion rules),
- $a \rightarrow ba, b \in V, a \in V^*$ (left insertion rules).

Or even a more general and non directional context can be expressed in $M_i$ (context rules):

- $dac \rightarrow dbc, a, b \in V, c, d \in V^*$ (substitution rules),
- $abc \rightarrow ac, b \in V, a, c \in V^*$ (deletion rules),
- $ac \rightarrow abc, b \in V, a, c \in V^*$ (insertion rules),

- $A_i$ is a finite set of strings over $V$. The set $A_i$ is the set of initial strings in the $i$-th node. Actually, in what follows, we consider that each string appearing in a node of the net at any step has an arbitrarily large number of copies in that node, so that we shall identify multisets by their supports.

- $PI_i$ and $PO_i$ are subsets of $V$ representing the input and output filter respectively. These filters are defined by membership condition, namely a string $w \in V^*$ can pass the input filter (the output filter) if $\forall x \in PI_i, w = axb$ where $a, b \in V^*$ ($\forall x \in PO_i, w = axb$ where $a, b \in V^*$).

  We write $\rho_i(w) = true$, if $w$ can pass the input filter of the node processor $i$, and $\rho_i(w) = false$, otherwise. We write $\tau_i(w) = true$, if $w$ can pass the output filter of the node processor $i$ and $\tau_i(w) = false$, otherwise.

A NEP with some of these rules is denoted by $NEP_b$ (basic rules), $NEP_d$ (directional rules) and $NEP_c$ (context rules).

By a configuration of an NEP as above we mean an $n$-tuple $C = (L_0, L_1, \cdots, L_n)$, with $L_i \subseteq V^*$ for all $0 \leq i \leq 6$. A configuration represents the sets of strings (remember that each string appears in an arbitrarily large number of copies) which are present in any node at a given moment; clearly the initial configuration of the network is $C_0 = (A_1, A_2, \cdots, A_n)$.

### 2.1 Dynamics of MPNEP

A configuration can change either by an evolutionary step or by a communicating step. Computation steps can be defined in a controlled way, that is, first an evolutionary step and then a communicating step; or in a parallel way, that is, evolution and communication take place at the same time.

When changing by an evolutionary step, each component $L_i$ of the configuration is changed in accordance with the evolution rules associated with the node $i$. Formally, we say that the configuration $C_1 = (L_1, L_2, \cdots, L_n)$ directly changes for the configuration $C_2 = (L'_1, L'_2, \cdots, L'_n)$ by an evolutionary step, written as

$$C_1 \Rightarrow C_2$$

if $L'_i$ is the set of strings obtained by applying the rules of $R_i$ to the strings in $L_i$ as follows:

- If the same substitution rule may replace different occurrences of the same symbol within a string, all these occurrences must be replaced within different copies of that string. The result is the multiset in which every string that can be obtained appears in an arbitrarily large number of copies.

- Unlike their common use, deletion and insertion rules are applied only to the end of the string. Thus, a deletion rule $a \rightarrow \varepsilon$ can be applied only to a string which ends by $a$, say $wa$, leading to the string $w$, and an insertion rule $\varepsilon \rightarrow a$ applied to a string $x$ consists of adding the symbol $a$ to the end of $x$, obtaining $xa$. If context rules are used, that is $ab \rightarrow a$ or $abc \rightarrow ac$ then the deletion point is defined by context information $b$ or $ac$.

- If more than one rule, no matter its type, applies to a string, all of them must be used for different copies of that string.

More precisely, since an arbitrarily large number of copies of each string is available in every node, after a evolutionary step in each node one gets an arbitrarily large number of copies of any string which can be obtained by using any rule in the set of evolution rules associated with that node. By definition, if $L_i$ is empty for some $0 \leq i \leq 6$, then $L'_i$ is empty as well.

When changing by a communicating step, each node processor sends all copies of the strings it has which are able to pass its output filter to all the other node processors and receives all copies of the strings sent by any node processor providing that they can pass its input filter.

Formally, we say that the configuration $C_1 = (L_1, L_2, \cdots, L_n)$ directly changes for the configuration
\[ C_2 = (L'_1, L'_2, \ldots, L'_n) \text{ by a communication step, written as} \]
\[ C_1 \vdash C_2 \]
if for every \(0 \leq i \leq n,\)
\[ L'_i = L_i \setminus \{ w \in L_i | \tau(w) = \text{true} \} \cup \bigcup_{j=0, j \neq i} \{ x \in L_j | \tau_j(x) = \text{true} \land p_i(x) = \text{true} \} \]

A parallel computation step among two configurations \(C_1\) and \(C_2\), represented by \(C_1 \vdash C_2\), if the application of the evolutionary and communication steps in parallel, that is,
\[ C_1 \vdash C_2 = (C_1 \vdash C_2) || (C_1 \Rightarrow C_2) \]

Let \(\Gamma = (V, N_1, N_2, \ldots, N_n)\) be an NEP. By a parallel computation in \(\Gamma\) we mean a sequence of configurations \(C_0, C_1, C_2, \ldots, C_t\), where \(C_0\) is the initial configuration and \(C_i \vdash C_{i+1}\) for all \(i \geq 0\). MPNEP is a NEP with parallel computation – MPNEP is a NEP with parallel computation.

If the sequence is finite, we have a finite computation. The result of any finite computation is collected in a designated node called the output node. If one considers the output node of the network as being the node \(N_0\), and if \(C_0, C_1, C_2, \ldots, C_t\) is a computation, then the set of strings existing in the node \(N_0\) at the last step – the 0-th component of \(C_t\) – is the result of this computation. The time complexity of the above computation is the number of steps, that is \(t\).

### 2.2 MPNEP vs. NEP

Let \(\Gamma = (V, N_1, N_2, \ldots, N_n)\) be an \(\text{NEP}_{\Gamma}\). By a controlled computation in \(\Gamma\) we mean a sequence of configurations \(C_0, C_1, C_2, \ldots, C_t\), where \(C_0\) is the initial configuration and \(C_{2i} \Rightarrow C_{2i+1}\) and \(C_{2i+1} \Rightarrow C_{2i+2}\) for all \(i \geq 0\).

**Theorem 2.1.** Problems solved using a \(\text{NEP}_{\Gamma}X\) can be solved using a MPNEP = \(\text{NEP}_{\GammaX}\), where \(X = \{b, d, c\}\).

**Proof.** Given a processor \(N_i = \{A, R, P, PO_i\}\) belonging to a \(\text{NEP}_{\GammaX}\) it is possible to transform it on a processor \(N'_i = \{A, R', P', PO'_i\}\) that behaves in the same way within a MPNEP = \(\text{NEP}_{\GammaX}\) in the following way:

- Given a rule \(r_{ik} \in R_i\) with the notation \(A \rightarrow B, \text{ with}\)
  \(1 \leq k \leq p\), each rule \(r'_{ik} \in R'_i\) has the form \(A \rightarrow BX_{ik}\)
- Given the output filter \(PO_i, PO'_i = \text{PO}_i \bigcup_{k=1}^p X_{ip}\)

With these new sets \(R'_i\) and \(\text{PO}'_i\) the parallel computation of a \(\text{NEP}\) behaves in the same way that a controlled computation since \(\tau(w) = \text{false}\) until all rules are applied.

**Theorem 2.2.** Problems solved using a \(\text{NEP}_{\GammaX}\) can be solved using a \(\text{NEP}_{\Gammad}\) and problems solved using a \(\text{NEP}_{\Gammac}\) can be solved using a \(\text{NEP}_{\Gammac}\), where \(X = \{p, c\}\).

**Proof.** It is obvious that \(abc \rightarrow ad\) where \(a, (c = \varepsilon) \in V^*\) is similar to \(ab \rightarrow ad\) and \(ab \rightarrow ad\) where \(a = \varepsilon\) is similar to \(b \rightarrow d\).

### 3 COMPUTATIONAL COMPLETE NEPS

Since MPNEP with finite filters can generate regular languages only, we shall consider in the sequel MPNEP having infinite regular languages as filters.

**Theorem 3.1.** Each recursively enumerable language can be generated by a complete MPNEP of size 5.

**Proof.** Let \(G = (N, T, S, P)\) be an arbitrary phrase-structure grammar in the Kuroda normal form, namely \(P\) contains only rules of the following forms:

\(A \rightarrow a, A \rightarrow BC, AB \rightarrow CD, A \rightarrow \varepsilon\)

where \(A, B, C, D\) are nonterminals and \(a\) is a terminal. We assume that the rules \(A \rightarrow BC\) and \(AB \rightarrow CD\) of \(P\) are labelled in a one-to-one manner by the labels \(r_1, r_2, \ldots, r_n\). We shall refer to the rules of the form \(A \rightarrow a, A \rightarrow \varepsilon, A \rightarrow BC, y AB \rightarrow CD\) as rules of type \(0, 1, 2,\) and \(3\), respectively. We construct the following NEP of size 5 having a complete underlying graph:

\(\Gamma = (N \cup T \cup V \cup \{X\}, N_0, N_1, N_2, N_3, N_4, K_5)\)

where \(V = \{r_i, p_i, q_i, s_i, t_i | 1 \leq i \leq n\}\) and

\(N_0 = (\{0, 0, T^\ast, (N \cup T \cup V \cup \{X\})^\ast (N \cup V \cup \{X\})^\ast \}
\(N \cup T \cup V \cup \{X\})^\ast \}
\(N_1 = (M_1, \{S\}, \{N \cup T\})^\ast \cup (N \cup T)^\ast \}
\(\{r_i q_i | 1 \leq i \leq n\} \cup (N \cup T)^\ast \}
\(N \cup T \cup \{X\})^\ast \}
\(\{N \cup T\}) \cup (N \cup T)^\ast \}

with \(M_1 = \{A \rightarrow X | A \rightarrow \varepsilon \in P\}
\cup \{A \rightarrow a | a \in P\} \cup \{A \rightarrow r_i, r_i \rightarrow t_i | r_i : A \rightarrow BC \in P\} \cup \{A \rightarrow s_i, B \rightarrow p_i | r_i : AB \rightarrow CD \in P\} \cup \)}
\[ N_2 = \{ (e \rightarrow q_i | r_i : A \rightarrow BC), \theta, \\
(\mathcal{N} \cup \mathcal{T})^* \{ r_i | 1 \leq i \leq n \} (\mathcal{N} \cup \mathcal{T})^* \} \]

\[ N_3 = \{ (M_3, \theta, (\mathcal{N} \cup \mathcal{T})^* \{ s_i p_i t_i q_i | 1 \leq i \leq n \} (\mathcal{N} \cup \mathcal{T})^* \} \]

with

\[ M_3 = \{ t \rightarrow B, q_i \rightarrow C | r_i : A \rightarrow BC \in P \} \cup \{ s_i \rightarrow C, p_i \rightarrow D | r_i : AB \rightarrow CD \in P \} \]

\[ N_4 = \{ (X \rightarrow e), \theta, (\mathcal{N} \cup \mathcal{T})^* \{ X \} (\mathcal{N} \cup \mathcal{T})^* \} \]

Taking into account this NEP configuration, a MPNEP can be defined according to theorem 2.1.

- Given a rule \( r_{ik} \in R_i \) with the notation \( A \rightarrow B \), with \( 1 \leq k \leq p \), each rule \( r'_{ik} \in R_i' \) has the form \( A \rightarrow Bx \).
- Given the output filter \( PO_i, PO'_i = PO_i \cup \bigcup_{l=1}^{p} X_{i} \).

This MPNEP behaves in the same way that a NEP and therefore each recursively enumerable language can be generated.

4 3-COLORABILITY PROBLEM

**Theorem 4.1.** The "3-colorability problem" can be solved in \( O(m+n) \) time by a MPNEP of size \( 4m+1 \), where \( n \) is the number of vertices and \( m \) is the number of edges of the input graph.

**Proof.** Let \( G = (\{1,2,\ldots,n\}, \{e_1, e_2, \ldots, e_m\}) \) a graph assume that \( e_i = \{k_i, l_i\}, 1 \leq k_i \leq l_i \leq n, 1 \leq i \leq m \). We consider the alphabet \( U = V \cup V' \cup T \cup A \), where \( V = \{b, r, g\} \), \( T = \{a_1, a_2, \ldots, a_n\} \), and \( A = \{\hat{A}_1, \hat{A}_2, \ldots, \hat{A}_n\} \).

We construct the following processors of a massive parallel NEP.

- A generator processor:
  \[ N_0 = \{ \{a_1, a_2, \ldots, a_n\}, \]
  \[ \{a_i \rightarrow b\hat{A}_i, a_i \rightarrow r\hat{A}_i, a_i \rightarrow g\hat{A}_i | 1 \leq i \leq n \}, \]
  \[ \emptyset, \{\hat{A}_i | 1 \leq i \leq n\} \} \]

This processor generates all possible color combinations, solutions or not, to the problem. And it sends those strings to next processors.

- For each edge in the graph \( e_i = \{k_i, l_i\} \), we have 4 filtering processors (where \( i = \{k_i, l_i\} \)):
  \[ N_{e_i}^1 = \{ \emptyset, \{ g\hat{A}_i \rightarrow g'a_i, r\hat{A}_i \rightarrow r'a_i \}, \]
  \[ \{\hat{A}_i, \{g', r'\} \} \]

It is clear to see that we can build a MPNEP with previous nodes in such a way that \( N_0 \) generates all possible colored strings and then apply processors \( N_{e_1}^1, N_{e_2}^1, \ldots, N_{e_m}^1 \) to filter such strings for edge \( e_i \). Repeating such filtering process with the rest of edges gives a valid solution to the given problem.

A MPNEP with the above architecture can solve the 3-colorability problem of \( n \) cities with \( m \) edges.

For the first \( n \) steps, that are evolution ones when nothing is actually communicated, the strings will remain in \( N_0 \) until no letter in \( T \) appears in them anymore. When this process is finished, the obtained strings encode all possible ways of coloring the vertices, satisfying or not the requirements of the problem. After this, step 1 is needed to communicate all possible solutions to next processors. Now, for each edge \( e_i \), MPNEP keeps only those strings which encodes a kol-colorability satisfying the condition for the two vertices of \( e_i \). This is done by means of the nodes \( N_{e_1}^2, N_{e_2}^2, \ldots, N_{e_m}^2 \), finally \( N_{e_m}^2 \) in 12 steps. As one can see, the overall time of a computation is \( 12m + 3n + 1 \). We finish the proof by mentioning that the total number of rules is \( 18m + 3n + 1 \). In conclusion, all parameters of the network are of \( O(m+n) \) size.

5 FINAL REMARKS

The concept of network of evolutionary processors is based on mechanisms inspired from cell biology. A variation of these networks called massive parallel network of evolutionary processors is also studied. These networks consists of nodes which are very simple processors and are able to perform rules operations. These nodes are endowed with a filter which are defined by regular sets. Evolution and communication is done in a parallel way which seem to be close to the possibilities of biological implementation rather than NEPs since bio-operation, chemical reactions, mutations etc. happen in a parallel way since there is no inherent synchronization among them.

A new dynamic of NEP has been introduced in this paper. It has been proof that any given NEP can be model with an equivalent MPNEP, see theorem 2.1, and therefore the inherent computational com-
pleteness of NEP can be incorporated in MPNEP. This massive parallel way of operation is closer to biological operations than classical NEP.

The 3-colorability problem has been solved using a software tool that simulates a MPNEP according to theorem 4.1. This soft tool can be use to check other useful theorems (Castellanos et al., 2001) in NEP research area.

This paper has proposed a new dynamic of NEP with a high parallel and non-deterministic behavior, and therefore all research about NEP can be translated into MPNEP. For instance, MPNEP can be modified according to (Diaz et al., 2007) in order to move filters towards edges. Each edge is viewed as a two-way channel such that input and output filters coincide. Thus, the possibility of controlling the computation in such networks seems to be diminished. In spite of this these networks are still computationally complete (Castellanos et al., 2006). Also, another rules can be implemented in MPNEP to extend this computing model. In (Manea et al., 2007) one replaces the point mutations associated with each node by the missing operation mentioned above, that of splicing. This new processor is called splicing processor. This computing model, called accepting network of splicing processors (shortly ANSP), is similar to some extent to the test tube distributed systems based on splicing.

REFERENCES


Errico, L. and Jesshope, C. (1994). Towards a new architecture for symbolic processing. Artificial Intell...