# NETWORKS OF PICTURE PROCESSORS

Victor Mitrana

***Abstract***
*The goal of this work is to survey in a systematic and uniform way the main results regarding different computational aspects of networks of picture processors viewed as rectangular picture accepting devices. We first consider networks with evolutionary picture processors only and discuss their computational power as well as a partial solution to the picture matching problem. Two variants of these networks, which are differentiated by the protocol of communication, are also surveyed: networks with filtered connections and networks with polarized processors. Then we consider networks having both types of processors, i.e., evolutionary processors and hiding processors, and provide a complete solution to the picture matching problem. Several results which follow from this solution are then presented. Finally we discuss some possible directions for further research.*

## 1. Introduction

A picture (2-dimensional word) is a rectangular array of symbols over an alphabet. Picture languages defined by different mechanisms have been studied extensively in the literature. Two-dimensional matrix and array models describing pictures have been proposed in [18, 19, 21, 20]. On the other hand, models defining pictures that are connected arrays, but not necessarily rectangular, have been proposed as early as 70's [17] and a hierarchy of grammars for such languages was considered in [22]. A new model of recognizable picture languages, extending to two dimensions the characterization of the one-dimensional recognizable languages in terms of alphabetic morphisms of local languages, was introduced in [8]. Similar to the string case, characterizations of recognizable picture series were proposed, see, e.g. [6, 14]. An early survey on automata recognizing rectangular picture languages is [10], a bit more recent one considering different mechanisms defining picture languages, not necessarily rectangular, is [17] and an even more recent and concise one is [9].

The main idea of the seminal work [5] in this area was to extend the investigation started in [12], where data is organized in the form of linear strings, to rectangular pictures. In [5], networks of evolutionary picture processors (ANEPP) where each node is either a row/column

substitution node or a row/column deletion node are considered. The action of each node on the data it contains is precisely defined. For instance, if a node is a row substitution node, then it can substitute a letter by another letter in either the top row only, the bottom row only, or an arbitrary row. Moreover, if there are more occurrences of the letter to be substituted in the row on which the substitution rule acts, then each such occurrence is substituted in different copies of that picture. An implicit assumption is that arbitrarily many copies of every picture are available. A similar informal explanation concerns the column substitution and deletion nodes. Local data is then transmitted over the network following a well defined protocol. Data can be communicated only if they pass a filtering process regulated by input and output filters (defined by some very simple context conditions) associated with each node. All the nodes simultaneously send their data to, and receive data from, the nodes they are connected to. In [5] we showed that these networks can accept the complement of any local language, as well as languages that are not recognizable.

In [3] one simplifies the ANEPP model considered in [5] by moving the filters from the nodes to the edges. A similar investigation for ANEPs has been initiated in [7] and continued in [2], where it was shown that both devices equal the computational power of Turing machines. Each edge of a network of evolutionary picture processors with filtered connections (ANEPPFC for short) is viewed as a two-way channel such that the input and output filters, respectively, of the two nodes connected by the edge coincide. Clearly, the possibility of controlling the computation in such networks seems to be diminished. For instance, there is no possibility to lose data during the communication steps. In spite of this fact all the results reported in [5] have been extended to these new devices. Moreover, in all cases the ANEPPFCs have a smaller size (number of processors).

A continuation of the aforementioned works is [1], where one considers the pattern matching problem, which is largely motivated by different aspects in low-level image processing [16], and tries to solve it in a parallel and distributed way with networks of picture processors. The network solving the problem can be informally described as follows: it consists of two subnetworks, one of them extracts at each step, simultaneously, all subpictures of identical (progressively decreasing) size from the input picture and sends them to the other subnetwork. In turn, this subnetwork consists of two subnetworks; one of them checks whether any of the received pictures is identical to the pattern, while the other one halts the computation if none of the received pictures is identical to the pattern. If the pattern is of size $(k, l)$, with $1 \leq k \leq 3$, and $l \geq 1$, we present an efficient solution running in $\mathcal{O}(n + m + l)$ computational (processing and communication) steps, provided that the input picture is of size $(n, m)$. Moreover, this solution can be extended at no further cost w.r.t. the number of computational steps to any finite set of patterns, all of the same size. From the proofs of these results we infer that any $(k, l)$-local language with $1 \leq k \leq 3$ can be decided in $\mathcal{O}(n + m + l)$ computational steps by networks with evolutionary processors. Particularly, every local language ca be decided in $\mathcal{O}(n + m)$ computational steps.

A new operation together with its inverse, that can convert a visible row/column into an invisible one and vice versa is also introduced in [1]. The two operations are called *mask* and *unmask*, respectively. We show how this variant of networks of picture processors is able to solve efficiently (in $\mathcal{O}(n + m + kl)$ computational steps) the problem of pattern matching of an

arbitrary pattern of size $(k,l)$ in a given rectangular picture of size $(n,m)$. Again, the solution can be extended at no further cost w.r.t. the number of computational steps to any finite set of patterns all of them of the same size. From the proofs of these results we infer that any $(k,l)$-local language with arbitrary $k,l$ can be decided in $\mathcal{O}(n+m+kl)$ computational steps by networks containing both evolutionary and hiding processors.

It is worth mentioning here that the complexity results mentioned above are to be interpreted at a very high level, as we only count the number of evolutionary and communication steps without taking into consideration the inherent time of these steps. For instance, each processing step of a single processor makes all possible transformations in parallel, producing all possible results in one step. This involves the duplication and modification of all pictures currently in that processor, which means that such a step may involve an exponential amount of internal work.

## 2. Basic Definitions

The basic terminology and notations concerning two-dimensional languages are taken from [9]. The set of natural numbers from 1 to $n$ is denoted by $[n]$. The set of all finite subsets of a set $A$ is denoted by $2^A$. The cardinality of a finite set $A$ is denoted by $card(A)$. We shall often omit the braces for singleton sets. Let $V$ be an alphabet, $V^*$ the set of one-dimensional strings over $V$ and $\varepsilon$ the empty string. A *picture* (or a two-dimensional string) over the alphabet $V$ is a two-dimensional array of elements from $V$. We denote the set of all pictures over the alphabet $V$ by $V_*^*$, while the empty picture will be still denoted by $\varepsilon$. A two-dimensional language over $V$ is a subset of $V_*^*$.

Let $\pi$ be a picture in $V_*^*$; we denote the number of rows and the number of columns of $\pi$ by $\overline{\pi}$ and $|\pi|$, respectively. The pair $(\overline{\pi},|\pi|)$ is called *the size* of the picture $\pi$. The size of the empty picture $\varepsilon$ is obviously $(n,m)$ with $nm=0$. Note that the empty picture is actually the (equivalence) class of all pictures of size $(n,m)$ with $nm=0$. The set of all pictures of size $(m,n)$ over the alphabet $V$, where $m,n\geq 1$, is denoted by $V_m^n$. The symbol placed at the intersection of the $i$th row with the $j$th column of the picture $\pi$, is denoted by $\pi(i,j)$.

Let $\pi$ be a picture of size $(m,n)$ over $V$; for any $1\leq i\leq k\leq m$ and $1\leq j\leq l\leq n$ we denote by ${}^{[i,j]}\pi_{[k,l]}$ the *subpicture* of $\pi$ having its leftmost upper corner in $\pi(i,j)$ and rightmost lower corner in $\pi(k,l)$ (it starts and ends at $(i,j)$ and $(k,l)$ in $\pi$, respectively). For any $i>k$ or $j>l$, we set ${}^{[i,j]}\pi_{[k,l]}=\varepsilon$. Furthermore, we simply write $\pi$ instead of ${}^{[1,1]}\pi_{[m,n]}$.

We recall now some definitions from [1]. For any alphabet $V$ and a symbol $a\in V$, we denote by $\not{a}$ the *invisible* copy of $a$; furthermore, we set $\not{V}:=\{\not{a}|\ a\in V\}$. We say that a picture $\pi\in(V\cup\not{V})_m^n$ is *well defined* if there exist $1\leq i\leq k\leq m$ and $1\leq j\leq l\leq n$ such that all elements of ${}^{[i,j]}\pi_{[k,l]}$ are from $V$ and all the other elements of $\pi$ are from $\not{V}$. In this case, we say that ${}^{[i,j]}\pi_{[k,l]}$ is the *maximal visible subpicture* of $\pi$. A rather intuitive way to understand a well defined picture $\pi$ is to consider that some rows and/or columns on the border of $\pi$ are hidden but not deleted. Note that any picture over $V$ is a well defined picture. For the rest of

this paper, we deal with well defined pictures only. The minimal alphabet containing all visible symbols appearing in a picture $\pi$ is denoted by $alph(\pi)$.

We now define the evolutionary operations on pictures. These definitions appear in [5, 3, 1], but we prefer to follow [1], where they are given in a more general setting.

Let $V$ be an alphabet; a rule of the form $a \to b$, with $a, b \in V \cup \{\varepsilon\}$ is called an *evolutionary rule*. We say that a rule $a \to b$ is: a) a *substitution rule* if neither $a$ nor $b$ is $\varepsilon$; b) a *deletion rule* if $a \neq \varepsilon$, $b = \varepsilon$; c) an *insertion rule* if $a = \varepsilon$, $b \neq \varepsilon$. In this paper we shall ignore insertion rules because we want to process every given picture in a space bounded by the size of that picture. We denote the sets of substitution and deletion rules by $Sub_V = \{a \to b \mid a, b \in V\}$ and $Del_V = \{a \to \varepsilon \mid a \in V\}$, respectively. Given a rule $\sigma$ as above and a picture $\pi \in (V \cup \#)^{n}_{m}$, we define the following *actions* of $\sigma$ on $\pi$ following [5].

If $\sigma \equiv a \to b \in Sub_V$, then $\sigma^{\leftarrow}(\pi)$ is the set of all pictures $\pi'$ such that the following conditions are satisfied:

(1.) There exist $1 \leq u \leq v \leq m$ and $1 \leq s \leq t \leq n$ such that ${}^{[u,s]}\pi_{[v,t]}$ is the maximal visible subpicture of $\pi$.

(2.a.) There exists $u \leq i \leq v$ such that $\pi(i,s) = a$; then $\pi'(i,s) = b$, and $\pi'(j,l) = \pi(j,l)$ for all $(j,l) \in ([m] \times [n]) \setminus \{(i,s)\}$.

(2.b.) If the leftmost column of ${}^{[u,s]}\pi_{[v,t]}$ does not contain any occurrence of $a$, then $\sigma^{\leftarrow}(\pi) = \{\pi\}$.

Informally, $\sigma^{\leftarrow}(\pi)$ is the set of all pictures that can be obtained from $\pi$ by replacing an occurrence of $a$ by $b$ in the leftmost column of the maximal visible subpicture of $\pi$. Note that $\sigma$ is applied to all occurrences of the letter $a$ in the leftmost column of the maximal visible subpicture of $\pi$ in different copies of the picture $\pi$. We say that the rule $\sigma$ is applied to the leftmost column of the maximal visible subpicture of $\pi$.

In an analogous way, we define $\sigma^{\rightarrow}(\pi)$, $\sigma^{\uparrow}(\pi)$, $\sigma^{\downarrow}(\pi)$, and $\sigma^{+}(\pi)$ as the sets of all pictures obtained by applying $\sigma$ to the rightmost column, to the first row, to the last row, and to any column/row of the maximal visible subpicture of $\pi$, respectively.

If $\sigma \equiv a \to \varepsilon \in Del_V$, then $\sigma^{\leftarrow}(\pi)$ is the picture obtained from $\pi$ by deleting the $i-th$ column of $\pi$ provided that the maximal visible subpicture of $\pi$ starts at the position $(i,j)$ in $\pi$, for some $j$, and the $i-th$ column of $\pi$ contains an occurrence of $a$. If the leftmost column of the maximal visible subpicture of $\pi$ does not contain any occurrence of $a$, then $\sigma^{\leftarrow}(\pi) = \pi$. We say that the deletion rule $\sigma$ is applied to the leftmost column of the maximal visible subpicture of $\pi$.

Analogously, $\sigma^{\rightarrow}(\pi)$, $\sigma^{\uparrow}(\pi)$, and $\sigma^{\downarrow}(\pi)$ are the pictures obtained from $\pi$ by applying $\sigma$ to the rightmost column, to the first row, and to the last row of the maximal visible subpicture of $\pi$, respectively. Furthermore, $\sigma^{|}(\pi)$ $(\sigma^{-}(\pi))$ is the set of pictures obtained from $\pi$ by deleting an arbitrary column (row) containing an occurrence of $a$ from $\pi$. If more than one column (row) of $\pi$ contains $a$, then for each such column (row), there is a copy of $\pi$ in $\sigma^{|}(\pi)$ $(\sigma^{-}(\pi))$ having this column (row) deleted. If $\pi$ does not contain any occurrence of $a$, then $\sigma^{|}(\pi) = \{\pi\}(\sigma^{-}(\pi) =$

$\{\pi\}$).

For every rule $\sigma$, symbol $\alpha \in \{\leftarrow, \rightarrow, \uparrow, \downarrow, |, -, +\}$, and $L \subseteq (V \cup \#)^*_*$, we define the *$\alpha$-action of $\sigma$ on $L$* by $\sigma^\alpha(L) = \bigcup_{\pi \in L} \sigma^\alpha(\pi)$. Given a finite set of rules $M$, we define the *$\alpha$-action of $M$* on the picture $\pi$ and the language $L$ by:

$$M^\alpha(\pi) = \bigcup_{\sigma \in M} \sigma^\alpha(\pi) \quad \text{and} \quad M^\alpha(L) = \bigcup_{\pi \in L} M^\alpha(\pi),$$

respectively. In what follows, we shall refer to the rewriting operations defined above as *evolutionary picture operations* since they may be viewed as the 2-dimensional linguistic formulations of local gene mutations.

We now define a new operation on pictures and its inverse, namely *mask* and *unmask* that was introduced in [1]. Let $\pi$ be a picture of size $(m, n)$ over $V \cup \#$ and $a \in V$.

- $mask^{\leftarrow}(\pi)$ returns the picture obtained from $\pi$ by transforming all visible symbols from the leftmost column of the maximal visible subpicture of $\pi$ into their invisible copies. Analogously, one defines the mappings $mask^{\rightarrow}$, $mask^{\uparrow}$, and $mask^{\downarrow}$.
- $unmask^{\leftarrow}(\pi)$ returns the picture obtained from $\pi$ as follows. If ${}^{[i,j]}\pi_{[k,l]}$ is the maximal visible subpicture of $\pi$, then all invisible symbols $\pi(s, j-1)$, $i \leq s \leq k$, become visible. If $j = 1$, then $unmask^{\leftarrow}(\pi) = \pi$. Analogously, one defines the mappings $unmask^{\rightarrow}$, $unmask^{\uparrow}$, and $unmask^{\downarrow}$.

For every $\alpha \in \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$ and $L \subseteq (V \cup \#)^*_*$, we define $mask^\alpha(L) = \{mask^\alpha(\pi) \mid \pi \in L\}$. Analogously, $unmask^\alpha(L) = \{unmask^\alpha(\pi) \mid \pi \in L\}$.

For two disjoint subsets $P$ and $F$ of an alphabet $V$ and a picture $\pi$ over $V$, we consider the following two predicates which we will later use to define two types of filters:

$$rc_s(\pi; P, F) \equiv P \subseteq alph(\pi) \wedge F \cap alph(\pi) = \emptyset$$
$$rc_w(\pi; P, F) \equiv alph(\pi) \cap P \neq \emptyset \wedge F \cap alph(\pi) = \emptyset.$$

The construction of these predicates is based on *context conditions* defined by the two sets $P$ (*permitting contexts/symbols*) and $F$ (*forbidding contexts/symbols*). Informally, both conditions require that no forbidding symbol is present in $\pi$; furthermore the first condition requires all permitting symbols to appear in $\pi$, while the second one requires that at least one permitting symbol appears in $\pi$.

For every picture language $L \subseteq V^*_*$ and $\beta \in \{s, w\}$, we define:

$$rc_\beta(L, P, F) = \{\pi \in L \mid rc_\beta(\pi; P, F) = \texttt{true}\}.$$

An *evolutionary picture processor with filters* over $V \cup \#$ is a 5-tuple $(M, PI, FI, PO, FO)$, where:

- Either $M \subseteq Sub_V$ or $M \subseteq Del_V$. The set $M$ represents the set of evolutionary rules of the processor. As one can see, a processor is "specialized" into one type of evolutionary operation, only.
- $PI, FI \subseteq V$ are the *input* sets of permitting and forbidding symbols (contexts) of the processor, while $PO, FO \subseteq V$ are the *output* sets of permitting and forbidding symbols of the processor (with $PI \cap FI = \emptyset$ and $PO \cap FO = \emptyset$).

An *evolutionary picture processor without filters* over $V \cup \boxplus$ is just a set of evolutionary rules.

A *hiding picture processor* over $V \cup \boxplus$ ([1]) is a 5-tuple $(M,\ PI, FI, PO, FO)$, where $M$ is either *mask* or *unmask*, while the other parameters are identical to those defined above for evolutionary processors.

# 3. Networks of Evolutionary Picture Processors

We give here the definition of an ANEPP following [1], which slightly differs from the definitions in [5] in the sense that the nodes $\underline{Halt}$ and $\underline{Accept}$ coincide. This definition is more suitable for ANEPP used as problem solvers.

An *accepting network of evolutionary picture processors* (ANEPP) is a 9-tuple

$$\Gamma = (V, U, G, N, \alpha, \beta, \underline{In}, \underline{Halt}, \underline{Accept}),$$

where:

- $V$ and $U$ are the input and network alphabet, respectively, $V \subseteq U$.
- $G = (X_G, E_G)$ is an undirected graph without loops with the set of vertices $X_G$ and the set of edges $E_G$. $G$ is called the *underlying graph* of the network. Although in network theory, several types of graphs are common like *complete, rings, stars, grids*, we focus here on complete underlying graphs (every two vertices are connected by an edge), so that we can replace the graph $G$ by the set of its nodes.
- $N$ is a mapping which associates with each node $x \in X_G$ the evolutionary picture processor with filters

$$N(x) = (M_x, PI_x, FI_x, PO_x, FO_x).$$

- $\alpha : X_G \longrightarrow \{\leftarrow, \rightarrow, \uparrow, \downarrow, |, -, +\}$; $\alpha(x)$ gives the action mode of the rules of node $x$ on the pictures existing in that node.
- $\beta : X_G \longrightarrow \{s, w\}$ defines the type of the *input* and *output* filters of a node. More precisely, for every node, $x \in X_G$, the following filters are defined:

  input filter: $\rho_x(\cdot) = rc_{\beta(x)}(\cdot; PI_x, FI_x)$,
  output filter: $\tau_x(\cdot) = rc_{\beta(x)}(\cdot; PO_x, FO_x)$.

  That is, $\rho_x(\pi)$ (resp. $\tau_x(\pi)$) indicates whether or not the picture $\pi$ can pass the input (resp. output) filter of $x$. More generally, $\rho_x(L)$ (resp. $\tau_x(L)$) is the set of pictures of $L$ that can pass the input (resp. output) filter of $x$.

- $\underline{In}, \underline{Halt}, \underline{Accept} \in X_G$ are the *input* node, the *halting* node, and the *accepting* node of $\Gamma$, respectively. Of course, it is not obligatory that the three nodes are different from each other.

We then say that $card(X_G)$ is the size of $\Gamma$. A *configuration* of an ANPP $\Gamma$ as above is a mapping $C : X_G \longrightarrow 2^{U_*^*}$ which associates a finite set of pictures with every node of the graph. A configuration may be understood as the sets of pictures which are present in any node at a given moment. Given a picture $\pi \in V_*^*$, the initial configuration of $\Gamma$ on $\pi$ is defined by $C_0^{(\pi)}(\underline{In}) = \{\pi\}$ and $C_0^{(\pi)}(x) = \emptyset$ for all $x \in X_G \setminus \{\underline{In}\}$.

A configuration can change via either a *processing step* or a *communication step*. When changing via a processing step, each component $C(x)$ of the configuration $C$ is changed in accordance with the set of rules $M_x$ associated with the node $x$ and the way of applying these rules, namely $\alpha(x)$. Formally, we say that the configuration $C'$ is obtained in *one processing step* from the configuration $C$, written as $C \Longrightarrow C'$, iff

$$C'(x) = M_x^{\alpha(x)}(C(x)) \text{ for all } x \in X_G.$$

When changing via a communication step, each node processor $x \in X_G$ sends one copy of each picture it has, which is able to pass the output filter of $x$, to all the node processors connected to $x$ (under our assumption, all nodes in $X_G$) and receives all the pictures sent by any node processor connected with $x$ provided that they can pass its input filter.

Formally, we say that the configuration $C'$ is obtained in *one communication step* from configuration $C$, written as $C \vdash C'$, iff

$$C'(x) = (C(x) \setminus \tau_x(C(x))) \cup \bigcup_{\{x,y\} \in E_G} (\tau_y(C(y)) \cap \rho_x(C(y))) \text{ for all } x \in X_G.$$

Note that pictures that cannot pass the output filter of a node remain in that node and can be further modified in the subsequent evolutionary steps, while pictures that can pass the output filter of a node are expelled. Further, all the expelled pictures that cannot pass the input filter of any node are lost.

Let $\Gamma$ be an ANPP; the computation of $\Gamma$ on an input picture $\pi \in V_*^*$ is a sequence of configurations $C_0^{(\pi)}, C_1^{(\pi)}, C_2^{(\pi)}, \ldots$, where $C_0^{(\pi)}$ is the initial configuration of $\Gamma$ on $\pi$, $C_{2i}^{(\pi)} \Longrightarrow C_{2i+1}^{(\pi)}$ and $C_{2i+1}^{(\pi)} \vdash C_{2i+2}^{(\pi)}$, for all $i \geq 0$. Note that configurations are changed by alternative steps. By the previous definitions, each configuration $C_i^{(\pi)}$ is uniquely determined by $C_{i-1}^{(\pi)}$. A computation as above *halts* if there exists a configuration such that the set of pictures existing in the halting node is non-empty. The *picture language decided* by $\Gamma$ is

$$L(\Gamma) = \{\pi \in V_*^* \mid \text{ the computation of } \Gamma \text{ on } \pi \text{ halts with a non-empty accepting node}\}$$

As we consider here ANPPs as problem solvers, the halting condition is a bit different than that from [5], where there is no deciding condition. Furthermore, for the rest of this paper we only

deal with ANPPs that halt on every input. These halting and deciding conditions are among several ways in which networks of evolutionary processors can be used as deciding machines in [11].

**Theorem 3.1** ([5])
*1. There exist non-recognizable languages which can be accepted by ANEPPs.*
*2. The complement of every local language can be accepted by an ANEPP.*

As a consequence of the results proved in [1] (and mentioned later), we can give the complexity of the membership problem for local languages.

**Theorem 3.2** *Every local language can be decided by ANEPPs in $\mathcal{O}(n+m)$ computational (processing and communication) steps.*

**Theorem 3.3** ([5]) *The class of languages accepted by ANEPP is closed under rotation, boolean union, projection, inverse projection.*

A natural problem is to find a pattern (a fixed picture) in a given picture. This problem is widely known as the two-dimensional pattern matching problem and is largely motivated by different aspects in low-level image processing [16]. The more general problem of picture matching (where it is not obligatory for the picture to be a two-dimensional array) is widely known in the Pattern Recognition field and is connected with Image Analysis and Artificial Vision [13, 23].

A key step in our solution is to construct a network able to decide the singleton language formed by a given picture.

**Theorem 3.4** *Let $\pi$ be a picture of size $(k,n)$ for some $1 \leq k \leq 3$ and $n \geq 1$. The language $\{\pi\}$ can be decided by an ANEPP.*

We give now a solution to the picture matching based on ANEPP, for patterns of size $(k,n)$ or $(n,k)$ for any $1 \leq k \leq 3$ and $n \geq 1$.

**Theorem 3.5** *Let $\pi$ be a picture of size $(k,l)$ for some $1 \leq k \leq 3$ and $l \geq 1$. The language*

$$\{\theta \mid \pi \text{ is a subpicture of } \theta\}$$

*can be decided by an ANEPP.*

It is worth mentioning that the construction used in the proof of the previously result (see [1]) can be easily extended to an ANEPP able to detect, in the same number of computational steps, any pattern from a finite set of pictures of the same size. It suffices to construct an independent subnetwork of the type just discussed for each pattern.

**Theorem 3.6** *Given a finite set $F$ of patterns of size $(k,l)$ and $(l,k)$ for all $1 \leq k \leq 3$ and $l \geq 1$, the pattern matching problem with patterns from $F$ can be solved by ANEPPs in $\mathcal{O}(n+m+l)$ computational (processing and communication) steps.*

However, this approach is not suitable for detecting patterns of a different size than those considered above. We give a complete solution to this problem in the section devoted to ANPP.

# 4. Networks of Evolutionary Picture Processors With Filtered Connections

It is clear that filters associated with each node allow a strong control of the computation. Indeed, every node has an input and output filter; two nodes can exchange data if it passes the output filter of the sender *and* the input filter of the receiver. Moreover, if some data is sent out by some node and not able to enter any node, then it is lost. In [3] the filters are moved from the nodes, as they are in [5], to the edges.

An *accepting network of evolutionary picture processors with filtered connections* (ANEPPFC) is a 10-tuple

$$\Gamma = (V, U, G, N, \alpha, R, \beta, \underline{In}, \underline{Halt}, \underline{Accept}),$$

where $V$, $U$, $G$, and $\alpha$ have the same meaning as in an ANEPP, while $N$ is a mapping which associates with each node $x \in X_G$ the evolutionary picture processor without filters $N(x) = (M_x)$. Furthermore, $R : E_G \longrightarrow 2^U \times 2^U$ is a mapping which associates with each edge $e \in E_G$ the disjoint sets $R(e) = (P_e, F_e)$, and $\beta : E_G \longrightarrow \{s, w\}$ defines the filter type of an edge.

As for an ANEPP, a configuration can change via either a processing step, which is defined exactly as the processing step in an ANEPP, or a communication step, which is defined as follows. Formally, we say that the configuration $C'$ is obtained in *one communication step* from configuration $C$, written as $C \vdash C'$, iff

$$C'(x) = (C(x) \setminus (\bigcup_{\{x,y\} \in E_G} rc_{\beta(\{x,y\})}(C(x), \mathcal{N}(\{x,y\}))))$$
$$\cup (\bigcup_{\{x,y\} \in E_G} rc_{\beta(\{x,y\})}(C(y), \mathcal{N}(\{x,y\})))$$

for all $x \in X_G$. The computation of $\Gamma$ on an input word, the halting and accepting conditions are the same to ANEPP.

The main results in [3] are:

**Theorem 4.1**
*1. There exist non-recognizable languages which can be accepted by ANEPPFCs.*
*2. The complement of every local language can be accepted by an ANEPPFC.*

In spite of the possible weakness of ANEPPFC, all the results reported in [5] have been extended to these new devices. Moreover, in all cases the ANEPPFCs have a smaller size (number of processors). This suggests that moving the filters from the nodes to the edges does not decrease the computational power of the model. It is worth mentioning that a direct proof showing that

ANEPs and ANEPs with filtered connections are computationally equivalent was proposed in [4]. However, that construction from [4] essentially need an operation, namely insertion, that has not a corresponding one in ANEPPs or ANEPPFCs, therefore we do not know a proof for a direct simulation of one model by the other.

# 5. Networks of Picture Processors

An *accepting network of picture processors* (ANPP) is defined as an ANEPPin which some nodes may be hiding picture processors. The computation, halting and accepting conditions are the same to those for ANEPP.

In the sequel, we show how the picture pattern matching can be completely solved with ANPP, that is with networks having both types of nodes: evolutionary processors and hiding processors. The idea is the same as in the case of ANEPP, namely the network solving the problem consists of two subnetworks, one of them extracts at each step, simultaneously, all subpictures of identical (progressively decreasing) size from the input picture and sends them to the other subnetwork. In turn, this subnetwork consists of two subnetworks; one of them checks whether any of the received pictures is identical to the pattern, while the other one halts the computation if none of the received pictures is identical to the pattern. Therefore, it suffices to construct an ANPP able to decide the singleton language formed by a given picture.

**Theorem 5.1** *Let $\pi$ be a picture of size $(k,l)$, for some $k,l \geq 1$ over an alphabet $V$. The language $\{\pi\}$ can be decided by an ANPP.*

We are now able to give the complete solution based on ANPPs to the problem of picture matching:

**Theorem 5.2** *Given a finite set $F$ of patterns of size $(k,l)$ and $(l,k)$ for any $k,l \geq 1$, the pattern matching problem with patterns from $F$ can be solved by ANPPs in $\mathcal{O}(n+m+kl)$ computational (processing and communication) steps.*

The solutions of picture matching with ANPP have several consequences (see [1]).

**Theorem 5.3** *Let $\pi$ be a picture of size $(k,n)$ for some $1 \leq k \leq 3$ and $n \geq 1$. The complement of the language $\{\pi\}$ can be decided by an ANEPP.*

**Theorem 5.4** *Let $(k,l)$ be two positive integers, $1 \leq k \leq 3$ and $l \geq 1$. Every $(k,l)$-local language or $(l,k)$-local language can be decided by ANEPPs in $\mathcal{O}(n+m+l)$ computational (processing and communication) steps.*

**Theorem 5.5** *Let $(k,l)$ be two positive integers. Every $(k,l)$-local language can be decided by ANPPs in $\mathcal{O}(n+m+kl)$ computational (processing and communication) steps.*

The last results lead to a similar solution to the following problem. Let $k, l$ be two positive integers and $F$ be a finite set of pictures of size $(k, l)$. The picture language $F_*^*$ is the minimal set of pictures such that:

$(i)$ $F \subseteq F_*^*$,

$(ii)$ If $\pi, \rho \in F_*^*$, then $\pi ® \rho \in F_*^*$ (provided that $\pi ® \rho$ exists) and $\pi © \rho \in F_*^*$ (provided that $\pi © \rho$ exists.)

**Theorem 5.6** *Let $k, l$ be two positive integers and $F$ be a finite set of pictures of size $(k, l)$. The language $F_*^*$ can be decided by ANPPs in $\mathcal{O}(n + m + kl)$ computational (processing and communication) steps.*

The networks including both evolutionary and hiding processors seem to be more powerful than ANEPPs considered in [5]. A natural further step is to investigate the computational power and other computational properties of ANPPs. For instance, the relationships between ANPPs and tiling systems are still unknown.

# 6. Networks of Polarized Evolutionary Picture Processors

The material presented here essentially follows [15]. A valuation is a mapping $\varphi : V \to \mathbb{Z}$ that associates to every symbol from $V$ an integer value. The value $\varphi(a)$ is also called the polarity of the symbol $a$. The *polarity of a picture* is denoted by the function $\Phi : V_*^* \to \{-, 0, +\}^4$ which associates a symbol from $\{-, 0, +\}$ to the top, bottom, leftmost and rightmost column, respectively. Informally we say that the polarity of a row or a column as being the *sign* of the sum of the polarities of every symbol in that row or column. The polarity of a picture can be seen as a 4-tuple over the set $\{-, 0, +\}$ and represents the polarities, in this order, of the first row, the last column, the last row and the first column. More formally, we define the polarity of a picture $\pi$, of size $(m, n)$ as:

$$\Phi(\pi) = (sign(\sum_{i=1}^{n} \varphi(\pi(1, i))), sign(\sum_{i=1}^{m} \varphi(\pi(i, n))), sign(\sum_{i=1}^{n} \varphi(\pi(m, i))), sign(\sum_{i=1}^{m} \varphi(\pi(i, 1)))).$$

A *polarized evolutionary picture processor* (ANPEPP) over $V$ is a 3-tuple $(M, \mathcal{AP}, \alpha)$ where:
-Either $M \subseteq Sub_V$ or $M \subseteq Del_V$ and represents the set of evolutionary rules associated with the processor.
-$\mathcal{AP} \in \{-, 0, +\}^4$ is a 4-tuple representing the picture polarity compatible with the processor
-$\alpha \in \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$ denotes the action mode of the rules of pictures associated with the processor.
We say a picture $\pi$ and a processor are *compatible* if and only if the picture's polarity, $\Phi(\pi)$, is identical to the $\mathcal{AP}$ element associated with the processor. We will denote the set of all polarized picture processors over the alphabet $V$ by $EPP_V$.

An *accepting network of polarized evolutionary picture processors* (ANPEPP for short) is like an ANEPP with all its processors being polarized picture processors.

As for an ANEPP, a configuration can change via either a processing step, which is defined exactly as the processing step in an ANEPP, or a communication step, which is defined as follows. Formally, we say that the configuration $C'$ is obtained in *one communication step* from configuration $C$, written as $C \vdash C'$, iff

$$C'(x) = (C(x) \backslash \{\pi \in C(x) | \Phi(\pi) \neq \mathcal{AP}_x\}) \cup \bigcup_{\{x,y\} \in E_G} (\{\pi \in C(y) | \Phi(\pi) = \mathcal{AP}_x\}).$$

The computation of $\Gamma$ on an input word, the halting and accepting conditions are the same to ANEPP.

As it was expected, the results from [5] are also obtained for this model:

**Theorem 6.1**
*1. There exist non-recognizable languages which can be accepted by ANPEP.*
*2. The complement of every local language can be accepted by an ANPEP.*

**Theorem 6.2** *Given a finite set $F$ of patterns of size $(k, l)$ and $(l, k)$ for all $1 \leq k \leq 3$ and $l \geq 1$, the pattern matching problem with patterns from $F$ can be solved by ANPEP.*

# 7. Final Remarks

The networks including both evolutionary and hiding processors seem to be more powerful than ANEPPs considered in [5]. A natural further step is to investigate the computational power and other computational properties of ANPPs. For instance, the relationships between ANPPs and tiling systems are still unknown.

Another direction of further research concerns the relationship between ANEPP and ANEPPFC. Remember that for the string case the two variants have initially been shown to be equivalent via simulations of Turing machines, and later on direct mutual simulations have been reported in [4]. Moreover, these simulations were shown to preserve the complexity. Are there possible similar simulations in the picture case?

The investigation started in [15] is just a first step in this direction. The communication protocol considered there seems very powerful. A natural step is to investigate more deeply the computational power of these networks. It would also be of interest if these networks can completely solve the picture pattern matching.

# References


[1] H. BORDIHN, P. BOTTONI, A. LABELLA, V. MITRANA, Networks of Picture Processors as Problem Solvers. *Soft Computing* (2016), DOI 10.1007/s00500–016–2206–y.

[2] P. BOTTONI, ET AL., Filter Position in Networks of Evolutionary Processors Does Not Matter. In: *DNA Based Computers*. LNCS 5877, Springer-Verlag, Berlin, 2009, 1–11.

[3] P. BOTTONI, ET AL., Networks of Evolutionary Picture Processors With Filtered Connections. In: *Unconventional Computation*. LNCS 5715, Springer-Verlag, Berlin, 2009, 70–84.

[4] P. BOTTONI, ET AL., Complexity-preserving Simulations Among Three Variants of Accepting Networks of Evolutionary Processors. *Natural Computing* 10 (2011), 429–445.

[5] P. BOTTONI, A. LABELLA, V. MITRANA, Networks of Evolutionary Picture Processors. *Fundamenta Informaticae* 131 (2014), 337–349.

[6] S. BOZAPALIDIS, A. GRAMMATIKOPOULOU, Recognizable Picture Series. *J. of Automata, Languages and Combinatorics* 10 (2005), 159–183.

[7] C. DRAGOI, F. MANEA, V. MITRANA, Accepting Networks of Evolutionary Processors With Filtered Connections. *J. UCS* 13 (2007), 1598–1614.

[8] D. GIAMMARRESI, A. RESTIVO, Recognizable Picture Languages. *Int. J. Pattern Recognition and Artificial Intelligence* 6 (1992), 241–256.

[9] D. GIAMMARRESI, A. RESTIVO, Two-dimensional Languages. In: G. ROZENBERG, A. SALOMAA (eds.), *Handbook of Formal Languages*. LNCS, Springer-Verlag, Berlin, 1997, 215–267.

[10] I. INOUE, I. TAKANAMI, A Survey of Two-dimensional Automata Theory. In: *Proc. 5th Int. Meeting of Young Computer Scientists*. LNCS 381, Springer-Verlag, Berlin, 1990, 72–91.

[11] F. MANEA, Complexity Results for Deciding Networks of Evolutionary Processors. *Theor. Comput. Sci.* 456 (2012), 65–79.

[12] M. MARGENSTERN, V. MITRANA, M. PEREZ-JIMENEZ, Accepting Hybrid Networks of Evolutionary Processors. In: *DNA Based Computers*. LNCS 3384, Springer-Verlag, Berlin, 2005, 235–246.

[13] K. MARRIOTT, B. E. MEYER, *Visual Language Theory*. Springer-Verlag Berlin, 1998.

[14] I. MAÜRER, Characterizations of Recognizable Picture Series. *Theor. Comput. Sci.* 374 (2007), 214–228.

[15] S. POPESCU, Networks of Polarized Evolutionary Picture Processors. *Romanian Journal of Information, Science and Technology* 18 (2015), 3–17.

[16] A. ROSENFELD, A. KAK, *Digital Picture Processing*. Academic Press, NY, 1982.

[17] A. ROSENFELD, R. SIROMONEY, Picture Languages – A Survey. *Languages of design* 1 (1993), 229–245.

[18] G. SIROMONEY, R. SIROMONEY, K. KRITHIVASAN, Abstract Families of Matrices and Picture Languages. *Computer Graphics and Image Processing* 1 (1972), 284–307.

[19] G. SIROMONEY, R. SIROMONEY, K. KRITHIVASAN, Picture Languages With Array Rewriting Rules. *Information and Control* 22 (1973), 447–470.

[20] K. SUBRAMANIAN, R. SIROMONEY, On Array Grammars and Languages. *Cybernetics and Systems* 18 (1987), 77–98.

[21] P. WANG, Sequential/Parallel Matrix Array Languages. *Journal of Cybernetics* 5 (1975), 19–36.

[22] P. WANG, Hierarchical Structure and Complexities of Parallel Isometric Patterns. *IEEE Trans. PAM I* 5 (1983), 92–99.

[23] P. WANG, H. B. (EDS.), *Handbook on Optical Character Recognition and Document Image Analysis*. World Scientific, 1996.