FILTERED NETWORKS OF EVOLUTIONARY PROCESSORS*

Luis Fernando de Mingo López, Eugenio Santos Menéndez,
Francisco Gisbert

Abstract: This paper presents some connectionist models that are widely 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. Neural Nets usually have a supervised or unsupervised learning stage in order to perform desired response. 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. This paper just shows some ideas about these two models.

Keywords: Natural Computation, Membrane Systems, Neural Networks, Networks of Evolutionary Processors.

ACM Classification Keywords: F.1.1 Models of Computation: Self-modifying machines (neural networks); F.4.1 Mathematical Logic: Computational logic

Introduction

Natural sciences, and especially biology, represented 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" identify new (unconventional) computational paradigms in different forms. There are attempts to define and investigate new mathematical or theoretical models inspired by nature, as well as investigations into defining programming paradigms that implement computational approaches suggested by biochemical phenomena. Especially since Adleman's experiment, 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.

In the last decade and especially after Adleman's experiment [1] a number of computational paradigms, inspired or gleaned from biochemical phenomena, are becoming of growing interest building a wealth of models, called generically Molecular Computing. New advances in, on the one hand, molecular and theoretical biology, and on the other hand, mathematical and computational sciences promise to make it possible in the near future to have accurate systemic models of complex biological phenomena. Recent advances in cellular Biology led to new models, hierarchically organized, defining a new emergent research area called Cellular Computing.

Numeric Models - Neural Networks

Neural networks are non-linear systems whose structure is based on principles observed in biological neuronal systems. A neural network could be seen as a system that can be able to answer a query or give an output as answer to a specific input. The in/out combination, i.e. the transfer function of the network is not programmed,

* Supported by INTAS 00-626 and TIC 2003-09319-c03-03.

but obtained through a ”training” process on empiric datasets. In practice the network learns the function that links input together with output by processing correct input/output couples. Actually, for each given input, within the learning process, the network gives a certain output that is not exactly the desired output, so the training algorithm modifies some parameters of the network in the desired direction. Hence, every time an example is input, the algorithm adjusts its network parameters to the optimal values for the given solution: in this way the algorithm tries to reach the best solution for all the examples. These parameters we are speaking about are essentially the weights or linking factors between each neuron that forms our network. Neural Networks' application fields are typically those where classic algorithms fail because of their inflexibility (they need precise input datasets). Usually problems with imprecise input datasets are those whose number of possible input datasets is so big that they can't be classified. For example in image recognition are used probabilistic algorithms whose efficiency is lower than neural networks' and whose characteristics are low flexibility and high development complexity. Another field where classic algorithms are in troubles is the analysis of those phenomena whose mathematical rules are unknown. There are indeed rather complex algorithms which can analyse these phenomena but, from comparisons on the results, it comes out that neural networks result far more efficient: these algorithms use Fourier's transform to decompose phenomena in frequential components and for this reason they result highly complex and they can only extract a limited number of harmonics generating a big number of approximations. A neural network trained with complex phenomena's data is able to estimate also frequential components, this means that it realizes in its inside a Fourier's transform even if it was not trained for that! One of the most important neural networks' applications is undoubtfully the estimation of complex phenomena such as meteorological, financial, socio-economical or urban events. Thanks to a neural network it's possible to predict, analyzing historical series of datasets just as with these systems but there is no need to restrict the problem or use Fourier's transform. A defect common to all those methods it's to restrict the problem setting certain hypothesis that can turn out to be wrong. We just have to train the neural network with hystorical series of data given by the phenomenon we are studying. Calibrating a neural network means to determinate the parameters of the connections (synapsis) through the training process. Once calibrated there is need to test the netowrk efficiency with known datasets, which has not been used in the learning process. There is a great number of Neural Networks which are substantially distingushed by: type of use, learning model (supervised/non-supervised), learning algorithm, architecture, etc.

Multilayer perceptrons (MLPs) are layered feed forward networks typically trained with static backpropagation. These networks have found their way into countless applications requiring static pattern classification. Their main advantage is that they are easy to use, and that they can approximate any input-output map. In principle, backpropagation provides a way to train networks with any number of hidden units arranged in any number of layers. In fact, the network does not have to be organized in layers any pattern of connectivity that permits a partial ordering of the nodes from input to output is allowed. In other words, there must be a way to order the units such that all connections go from ”earlier” (closer to the input) to ”later” ones (closer to the output). This is equivalent to stating that their connection pattern must not contain any cycles. Networks that respect this constraint are called feed forward networks; their connection pattern forms a directed acyclic graph or dag.

Jordan and Elman networks extend the multilayer perceptron with context units, which are processing elements that remember past activity. Context units provide the network with the ability to extract temporal information from the data. In Elman networks, the activity of the first hidden layer are copied to the context units, while the Jordan network copies the output of the network. Jordan and Elman networks combine the past values of the context unit with the present input x to obtain the present net output. The Jordan context unit acts as a so called lowpass filter, which creates an output that is the weighted (average) value of some of its most recent past outputs.

Time lagged recurrent networks are MLPs extended with short term memory structures. Most real-world data contains information in its time structure. Yet, most neural networks are purely static classifiers. TLRNs are the state of the art in nonlinear time series prediction, system identification and temporal pattern classification.

Symbolic Models - Cellular Computing

P-systems represent a class of distributed and parallel computing devices of a biological type that was introduced in [14] which are included in the wider field of cellular computing. Several variants of this model have been investigated and the literature on the subject is now rapidly growing. The main results in this area show that P-systems are a very powerful and efficient computational model [15], [16], [13]. There are variants that might be classified according to different criteria. They may be regarded as language generators or acceptors, working with strings or multisets, developing synchronous or asynchronous computation. Two main classes of P-systems can be identified in the area of membrane computing [15]: cell-like P-systems and tissue-like P-systems. The former type is inspired by the internal organization of living cells with different compartments and membranes hierarchically arranged; formally this structure is associated with a tree. Tissue P-systems have been motivated by the structure and behaviour of multicellular organisms where they form a multitude of different tissues performing various functions [2]; the structure of the system is instead represented as a graph where nodes are associated with the cells which are allowed to communicate alongside the edges of the graph.

More recently, a notion of population P-systems has been introduced [3], [4] as a model for tissue P-systems where the structure of the underlying graph can be modified during a computation by varying the set of nodes and the set of edges in the graph. Specifically, nodes are associated with cells, each of them representing a basic functional unit of the system, and edges model bonds among these cells that are dynamically created and destroyed. Although mainly inspired by the cell behavior in living tissues, population P-systems may be also regarded as an abstraction of a population of bio-entities aggregated together in a more complex bio-unit (e.g. social insects like ants, bees, wasps etc, organized in colonies or bacteria of different types). This is the main reason why we use the term population instead of tissue albeit the term cell is retained to denoting an individual in the system. The concept also recalls other similar computational models: grammar systems [8], eco-grammar systems [9], or more recently, networks of parallel/evolutionary processors [10].

Universality results have been obtained [4] for a number of variants of population P-systems. The following different rules are considered: transformation rules for modifying the objects that are present inside the cells, communication rules for moving objects from a cell to another one, cell division rules for introducing new cells in the system, cell differentiation rules for changing the types of the cells, and cell death rules for removing cells from the system. As well as this, bond-making rules are considered that are used to modify the links between the existing cells (i.e., the set of edges in the graph) at the end of each step of evolution performed by means of the aforementioned rules. In other words, a population P-system in [4] is basically defined as an evolution-communication P-system [7] but with the important difference that the structure of the system is not rigid and it is represented as an arbitrary graph. In particular, bond making rules are able to influence cell capability of moving objects from a place to another one by varying the set of edges in the underlying graph.

Another interesting variant of population P-systems is obtained by considering the general mechanism of cell communication based on signal molecules as a mechanism for triggering particular transformations inside of a cell once a particular signal-object has been received from some other cell in the system [3]. This leads to a notion of population P-systems where the sets of rules associated with the cell can vary according to the presence of particular objects inside and outside the cells. Yet again, the introduction of this mechanism is motivated by the features shared by biological systems at various levels where the behavior of an individual is affected both by its internal state and by the external stimuli received. Some results concerning the power of population P-systems with a rule activating mechanism have been obtained [5].

Further developments of the area of population P-systems are expected to cover alternative ways of defining the result of a computation and the use of string objects. Population P-systems in fact attempt to model aspects of biological systems formed by many different individual components cooperating in a coherent way for the benefit

of the system as a whole; a more appropriate notion of computation is therefore necessary in order to characterize the emergent behavior of the system. Existing approaches in the area of grammar system such parallel communicating grammar systems [8] or eco-grammar systems [9], rely on the use of a single sentential form that is rewritten in parallel by different interacting/cooperating grammar components. In particular, in the case of eco-grammar systems, this sentential form is associated with the environment and it can be rewritten both by rules corresponding to action taken from the individual components in the system and by dedicated rules associated with the environment. In a similar way, we can consider string-processing population P-systems where the result of a computation is given by a string (or a language) produced in the environment at the end of a computation. However, with respect to grammar systems, population P-systems present some other interesting features like the possibility of moving objects from a place to another one, the possibility of forming bonds among the cells, the possibility of introducing new cells in the system by means of cell division, which need to be formalized for the particular case of string objects. In this respect, we aim to present some reasonable variants of population P-systems with string objects.

Membranes in P-systems can be connected using a graph and all of them can be treated as skin ones forming a so called Network of Evolutionary Processors. A network of evolutionary processors of size n is a construct NEP = $(V, N_1, N_2,\dots, N_n, G)$, where V is an alphabet and for each $1 \le i \le n$, $N_i = (M_i, A_i, PI_i, PO_i)$ is the i-th evolutionary node 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:
  $a \prod b$, $a, b \in V$ (substitution rules)
  $a \prod \varepsilon$, $a \in V$ (deletion rules)
  $\varepsilon \prod a$, $a \in V$ (insertion rules)
  More clearly, the set of evolution rules of any processor contains either substitution or deletion or 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 any node 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 the output filter, respectively. These filters are defined by the membership condition, namely a string $w \in V^*$ can pass the input filter (the output filter) if $w \in PI_i$ ($w \in PO_i$).

$G = (N_1, N_2,\dots, N_n, E)$ is an undirected graph called the underlying graph of the network. The edges of G, that is the elements of E, are given in the form of sets of two nodes. The complete graph with n vertices is denoted by K_n. By a configuration (state) of an NEP as above we mean an n-tuple $C = (L_1, L_2,\dots, L_n)$\$, with $L_i \subseteq V^*$ for all $1 \le i \le n$. 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 $C_0 = (A_1, A_2,\dots, A_n)$.

A configuration can change either by an evolutionary step or by a communicating step. When changing by an evolutionary step, each component L_i of the configuration is changed in accordance with the evolutionary rules associated with the node i. When changing by a communication step, each node processor N_i sends all copies of the strings it has which are able to pass its output filter to all the node processors connected to N_i and receives all copies of the strings sent by any node processor connected with N_i providing that they can pass its input filter.

Theorem 1. Each recursively enumerable language can be generated by a complete NEP of size 5. [21]

Theorem 2. Each recursively enumerable language can be generated by a star NEP of size 5. [22]

Theorem 3. The bounded PCP can be solved by an NEP in size and time linearly bounded by the product of K and the length of the longest string of the two Post lists. [23]

A simple NEP of size n is a construct SNEP = $(V, N_1, N_2, \dots, N_n, G)$, where, V and G have the same interpretation as for NEPs, and for each $1 \leq i \leq n$, $N_i = (M_i, A_i, PI_i, FI_i, PO_i, FO_i)$ is the i-th evolutionary node processor of the network. M_i and A_i from above have the same interpretation as for an evolutionary node in a NEP, but:

- PI_i and FI_i are subsets of V representing the input filter. This filter, as well as the output filter, is defined by random context conditions, PI_i forms the permitting context condition and FI_i forms the forbidding context condition. A string $w \in V^*$ can pass the input filter of the node processor i, if w contains each element of PI_i but no element of FI_i. Note that any of the random context conditions may be empty, in this case the corresponding context check is omitted. We write ρ_i (w) = true, if w can pass the input filter of the node processor i and ρ_i (w) = false, otherwise.
- PO_i and FO_i are subsets of V representing the output filter. Analogously, a string can pass the output filter of a node processor if it satisfies the random context conditions associated with that node. Similarly, we write τ_i (w) = true, if w can pass the input filter of the node processor i and τ_i (w) = false, otherwise.

Theorem 4. The families of regular and context-free languages are incomparable with the family of languages generated by simple NEPs. [21]

Theorem 5. The "3-colorability problem" can be solved in O(m + n) time by a complete simple NEP of size 7m+2, where n is the number of vertices and m is the number of edges of the input graph. [21]

Filtered Connections in NEPs

Main idea in NEPs is based on the fact that filters are inside processors in order to control what objects can pass through connections, but these filters make complex proccessors. If such filters are in connections, instead in processors, the simplicity of processors will increase compare to classical NEPs.

A network of evolutionary processors with filtered connections of size n is a construct FNEP = $(V, N_1, N_2, \dots, N_n, G)$ where V is an alphabet and for each $1 < i < n$, $N_i=(M_i, A_i)$ is the i-th evolutionary node processor of the network. The parameters of every processor are:

- M_i is a finite set of evolution rules (substitution, deletion or insertion rules).
- A_i is a finite set of strings over V. The set A is the set of initial strings in the i-th node. Actually, in what follows, we consider that each string appearing in any node at any step has an arbitrarily large number of copies in that node, so that we shall identify multisets by their supports.

Finally, $G=(N_1, N_2, \dots, N_n, (E,F))$ is an directed graph called the underlying graph of the network. The edges of G, that is the elements of (E,F), are given in the form (e_i, f_i) where f_i is the filter associated to connection e_i. Elements in F are just object sets, an element w pass the filter in f_i if $w \in f_i$. The complete graph with n vertices is denoted by K_n. By a configuration (state) of an NEP as above we mean an n-tuple $C=(L_1, L_2, \dots, L_n)$, with $L_i \subseteq V^*$ for all $1 < i < n$. 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 $C_0 = (A_1, A_2, \dots, A_n)$.

A configuration can change either by an evolutionary step or by a communicating step. When changing by an evolutionary step, each component L_i of the configuration is changed in accordance with the evolutionary rules associated with the node i. When changing by a communication step, each node processor N_i sends all copies of the strings it has to all the node processors connected to N_i and receives all copies of the strings sent by any node processor connected with N_i (providing that all sent/received information pass filters in connections).

Theorem 6. Solved problems using NEPs can be solve using NEPs with filtered connections.

Given two processor of a NEP, N_i and N_j connected by edge e_{ij} and with filters (FI_i,FO_i) and (FI_j,FO_j), they can be transformed into two processors of a NEP with filtered connections in the following way:

- Remove filters from processors N_i and N_j to obtain processors of NEPFC M_i and M_j.
- Add the filter $f_{ij}=FO_i \wedge FI_j$ to a connection from processor M_i -> M_j.
- Add the filter $f_{ji}=FO_j \wedge FI_i$ to a connection from processor M_i <- M_j.

It is clear that this kind of transformation produces a NEPFC with the same behaviour than a NEP.

Theorem 7. A NEPFC with m processors and less than c=2m connections can not be transformed into an equivalent NEP.

Each c connection is an equation with two unknows, so there are 2c unknows (input and output filters in NEPs) and there exists 2m filters to compute. So if the c<2m the system has infinite solutions but the behaviour will not be the same in all cases. If c=2m the system has only one solution, and if c>2m the system has only one solution.

Therefore, if NEPs and NEPFCs are equivalent under some constraints then all theorems in NEPs are valid for NEPFC. This new model can solve NP-problems in linear time.

Conclusions

This paper has introduced the novel computational paradigm Networks of Evolutionay Processors. Connectionists models such as Neural Networks can be taken into account to develop NEP architecture in order to improve behaviour. As a future research, learning concepts in neural networks can be adapted in a NEP architecture provided the numeric-symbolic difference in both models. NEPs and NEPFCs can be consired universal models since the are able to solve NP-problems. The great disadvantage is that a given NEP/NEPFC can only solve a given problem, if it is necessary to solve another problem (maybe a little variation) then another different NEP/NEPFC has to be implemented. The idea of learning tries to undertake such disadvantage proposing a model able to solve different kinds of problems (that is a general class of problems). Learning proposed can be based on the self organizing maps. There are a lof of open problems that need to be solved in order to show the computational power of this model, but the possibility to compute NP-problems is promising apart from the massive parallelization and non-determinism of the model.

Bibliography

[1] Adleman, L.M. 1994. Molecular computation of solutions to combinatorial problems. Science, 226, 1021-1024

[2] Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P. 2002. The Molecular Biology of the Cell. Fourth Edition. Garland Publ. Inc., London

[3] Bernardini, F., Gheorghe, M. 2004. Cell Communication in Tissue Psystems and Cell Division in Population P-Systems. In [17], 74-91

[4] Bernardini, F., Gheorghe, M. 2004. Population P-Systems. Journal of Universal Computer Science, 10, 509-539

[5] Bernardini, F., Gheorghe, M. 2005. Cell Communication in Tissue P-Systems: Universality Results. Soft Computing (to appear)

[6] Busby, S., de Lorenzo, V. 2001. Cell regulation - putting together pieces of the big puzzle. Curr. Op. Microbiol, 4, 117-118.

[7] Cavaliere, M. 2003. Evolution Communication P-Systems. In [16], 134-145 and in [17], 206-223.

[8] Csuhaj-Varju, E., Dassow, J., Kelemen, J., Paun, Gh. 1997. Grammar Systems. A Grammatical Approach to Distribution and Cooperation. Gordon and Breach, London

[9] Csuhaj-Varju, E., Kelemen, J., Kelemenova, A., Paun, Gh. 1997. EcoGrammar Systems: A Grammatical Framework for Studying Life-Like Interactions. Artificial Life, 3, 1-28.

[10] Csuhaj-Varju, E., Salomaa, A., 1997. Networks of Parallel Language Processors. In New Trends in Formal Languages. Control, Cooperation, and Combinatorics. In Paun, Gh., Salomaa, A. (eds), Lecture Notes in Computer Science, 1218, Springer-Verlag, Berlin, Heidelberg, New York, 299-318

[11] Krasnogor, N., Gheorghe, M., Terrazas, G., Diggle, S., Williams, P., Camara, M. 2005. Bulletin of the EATCS (to appear)

[12] Jennings, N.R. 2000. On agent-based software engineering. Artificial Intelligence, 117, 277-296.

[13] Martin-Vide, C., Mauri, G., Paun, Gh., Rozenberg, G., Salomaa, A. (eds) 2004. Membrane computing. International workshop, WMC 2003, Tarragona, Spain, July 2003. Revised papers. Lecture Notes in Computer Science, 2933, Springer, Berlin Heidelberg New York

[14] Paun, Gh. 2000. Computing with Membranes. Journal of Computer and System Sciences, 61, 108-143

[15] Paun, Gh. 2002. Membrane computing. An introduction. Springer, Berlin Heidelberg New York

[16] Paun, Gh., Rozenberg, G., Salomaa, A., Zandron, C. (eds) 2003. Membrane computing. International workshop, WMC-CdeA 02, Curtea de Arges, Romania, August 19-23, 2002. Revised papers. Lecture Notes in Computer Science, 2597, Springer, Berlin Heidelberg New York

[17] Paun, Gh., Riscos-Nunez, A., Romero-Jimenez, A., Sancho-Caparrini, F. (eds) 2004. Second brainstorming week on membrane computing, Seville, 2-7 February 2004. Technical Report 01/2004, Department of Computer Science and Artificial Intelligence, University of Seville, Spain

[18] Swift, S., Downie, J.A., Whitehead, N.A., Barnard, A.M.L., Salmond, G.P.C., Williams, P. 2001. Quorum sensing as a population-densitydependent determinant of bacterial physiology. Adv Micro Physiol, 45, 199270.

[19] Williams, P., Camara, M., Hardman, A., Swift, S., Milton, D., Hope, V.J., Winzer, K., Middleton, B., Pritchard, D.I., Bycroft, B.W. 2000. Quorum sensing and the population dependent control of virulence. Phil. Trans Roy Soc London B, 355(1397), 667-680.

[20] Winzer, K., Hardie, K.H., Williams, P. 2002. Bacterial cell-to-cell communication: sorry can't talk now – gone to lunch!. Curr Op. Microbiol, 5, 216-222.

[21] Juan Castellanos, Carlos Martin-Vide, Victor Mitrana, Jose M. Sempere; *Networks of Evolutionary Processors*.

[22] Juan Castellanos, Carlos Martin-Vide, Victor Mitrana, Jose M. Sempere; *Solving NP-Complete Problems with Networks of Evolutionary Processors*. IWANN.

[23] Carlos Martin-Vide, Victor Mitrana, Mario J. Perez-Jimenez, Fernando Sancho-Caparrini; *Hybrid Networks of Evolutionary Processors*. GECCO 2003. Lecturne Notes on Computer Science 2723, pp. 401-412. 2003.

Authors' Information

Luis Fernando de Mingo López – Dept. Organización y Estructura de la Información, Escuela Universitaria de Informática, Universidad Politécnica de Madrid, Crta. De Valencia km. 7, 28031 Madrid, Spain; e-mail: lfmingo@eui.upm.es

Eugenio Santos Menéndez – Dept. Organización y Estructura de la Información, Escuela Universitaria de Informática, Universidad Politécnica de Madrid, Crta. De Valencia km. 7, 28031 Madrid, Spain; e-mail: esantos@eui.upm.es

Francisco Gisbert – Dept. Organización y Estructura de la Información, Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo, 28660 Boadilla del Monte, Madrid, Spain; e-mail: fgisbert@fi.upm.es