2019 IEEE International Conference on Big Data (Big Data)

Graph classification with the hypernetwork, a molecule interaction based evolutionary architecture

Jose Segovia-Juarez
Computer Science Department
National University of Engineering
Lima, Peru
jsegovia@uni.pe

Silvano Colombano
Intelligent Systems Division
NASA Ames Research Center
Moffett Field, CA 94035, USA
silvano.p.colombano@nasa.gov

Alex Flores-Mamani
Computer Science Department
National University of Engineering
Lima, Peru
alex.flores.m@uni.pe

Daniel Hidalgo-Chavez
Computer Science Department
National University of Engineering
Lima, Peru
dhidalgoc@uni.pe

Miguel Mejia-Puma
Computer Science Department
National University of Engineering
Lima, Peru
miguelmejia@uni.pe

***Abstract*—A novel architecture for information processing, called the hypernetwork architecture is described here. This model is based on the hierarchical organization and principles of biological information processing. The hypernetwork model has a representation of the molecular, cellular, and organismic levels of biological organization. Molecules are enzyme-like structures, and interactions are typical activation and inhibition processes. The representation of molecules and their interactions is comprised of binary strings and string matching respectively. Molecules are placed in cells, modeled by cellular automata. An organized group of cells forms an organism. Cell to cell interactions are produced by the effector-receptor molecules of the cells. The hypernetwork receives environmental influences at its input cells, creates cascades of molecular interactions inside the cells, passing through internal cells, and delivers an output from its output cells. Hypernetwork organisms learn classification tasks, including graph classification, by an adaptive algorithm based on molecular evolution. An organism is reproduced with random molecular mutation and the selection chooses the organism with the best structure for the problem to be solved. With its molecule based variation-selection learning algorithm, the hypernetwork is able to learn fairly complex classification tasks. Besides learning, the hypernetwork exhibits mutation buffering capabilities, intra-cellular feedback regulation, and can be used as a tool for understanding how hierarchies work, for studying evolutionary strategies, and as a model for building molecular computers.**

***Index Terms*—Graph Learning, hypernetwork, graph classification, subgraphs, evolutionary computing, molecule based variation selection algorithm.**

I. Introduction

The hypernetwork architecture is a biologically inspired learning model that comprises three hierarchical levels: organism, cellular, and molecular, and attempts to capture the following properties: cross-scale information flow [1], learning [2], [3], evolvability, and mutation-buffering [4], [5].

The hypernetwork architecture has three components:

1) The structure, consisting of molecular, cellular, and organismic levels.
2) The dynamics, based on the molecular interactions modeled after protein self-assembly forming networks of molecular interactions inside and between cells.
3) The learning algorithm, based on molecular mutation and selection.

The primary elements of the hypernetwork model are molecules and their interactions. In the model molecules are represented by binary strings, and their shape based interactions are modeled with string matching. Information processing within the cells is given by their internal dynamics, however the computation power is based on molecular self-assembly processes [6], represented by binary string complementarity matching. Sets of molecules of different types form cells. Cell to cell interactions that are based on receptor and effector molecules. External influences on receptor molecules of input cells percolate into the cells to trigger cascades of molecular reactions, expressed in the form of networks of molecular interactions. Intra-cellular dynamics are filtered up to higher levels to the organismic level. The output of the organism is obtained from the readout structures of output cells.

The adaptive (variation - selection) algorithm for learning is based on molecular evolution. An organism is reproduced with molecular mutation, and the best organism from the offspring is selected for the next iteration. The organism changes its molecular components, by changing its molecular structure, and consequently its potential molecular interactions. The organism with the best molecular structures will be selected, until the complete table is learned or a termination condition is fulfilled.

Applying hypernetwork learning to solve graph classification problems is strightforward. Graphs are encoded by binary strings to be fed the hypernetwork, which in turn, form interaction graphs (information flow) for each input graph, and the output is obtained as a binary vector (see Figure 1). The output vector could represent a class, or another graph.

Input	Algorithm	Output
Graph $\rightarrow$	Hypernetwork (Graph) $\rightarrow$	Graph/Class

Fig. 1. The hypernetwork generate graphs to classify graphs.

This paper introduces applications of the hypernetwork architecture for graph classification problems. Section II is a detailed description of the hypernetwork architecture. Section III describes the variation-selection algorithm based on molecular mutations. Section IV shows the the hypernetwork architecture addressing a five node graph classification task. Section V discusses the experimental results, key hypernetwork features, and points to future work. Finally, we state our conclusions in Section VI.

II. The hypernetwork architecture description

The hypernetwork architecture is described bottom up, from the molecular to the organismic levels. The model includes molecular, cellular, and organismic (or tissue) levels of organization.

A first view is shown in Figure 2. The system has an environment with which it interacts via input and outputs.

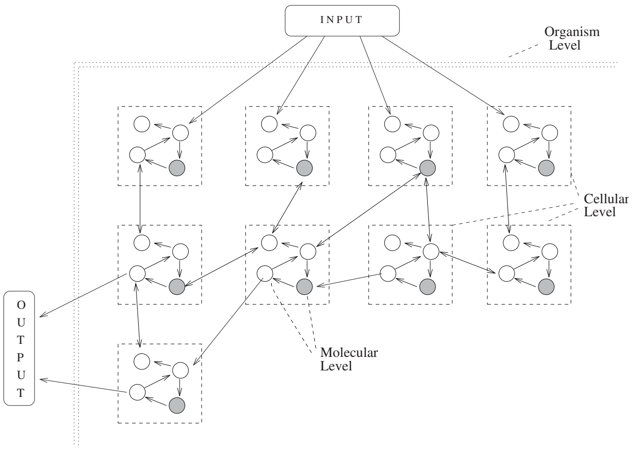


Fig. 2. Three hierarchical levels of the hypernetwork model: molecular, cellular, and organismic. The arrows show interactions, forming networks of interections. The system has an environment with which the hypernetwork interacts via input (graph inputs) and outputs (graph outputs).

A. Molecular level

The molecular level is modeled by molecules reminiscent of proteins. Molecules are represented by a binary string. The *self-assembly* of proteins and other macromolecules, "a self-organizing process driven by free energy minimization" [7], is modeled by *interactions* based on shape complementarity. Two molecules react if their recognition sites are complementary to each other.

The molecule has three parts, or domains, of a given length (usually 14 bits) each (see Figure 3 A), with different functions. There are two receptor domains, *excitatory* and *inhibitory*, that set the molecule into the active state or the inactive state, respectively. The third domain is called the *catalytic* domain, by means of which the molecule will activate neighbor molecules if there is complementarity matching (i.e., matching above a threshold) to the receptor domain of the target molecules [2]. The catalytic domain of a molecule can activate the target molecule by matching its excitatory receptor domain, or inhibit the target molecule by matching its inhibitory receptor domain as observed in Figure 3 B.

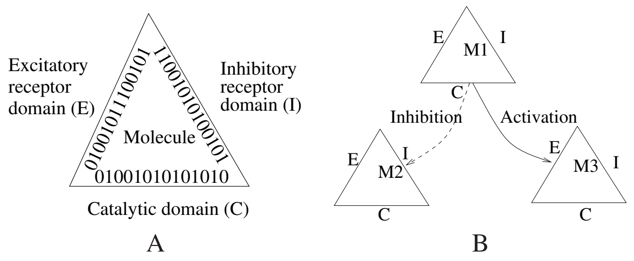


Fig. 3. (A) Representation of a molecule with inhibitory, excitatory and catalytic domains. (B) The catalytic domain of Molecule M1 interacts with the inhibitory receptor domain of M2, and the excitatory receptor domain of M3. The excitatory (E), inhibitory (I), and catalytic (C) domains of each molecule are shown.

The matching between the molecule domains is determined by the *Hamming distance matching rule* defined as follows: given two binary strings $x = x_1x_2x_3\ldots x_k$, and $y = y_1y_2y_3\cdots y_k$,

$$x \; matches \; y = \sum_i \overline{x_i \oplus y_i} \; \geq r, \tag{1}$$

where $\oplus$ is the exclusive-or operator, and $1 \leq r \leq k$ is a threshold value.

The existence of inhibitory and excitatory domains allows the creation of *positive* and *negative feedback regulatory interactions* with neighbor molecules.

There are many molecular types in an actual cell such a neuron. From them we identified three types of molecules, by their function: receptors, effectors, and internals.

- Receptors are molecules that are sensitive to external influences (from the environment or from molecules of other cells). They behave like membrane receptors in neurons.
- Effectors are molecules that transfer information out of a cell into other cells via their receptors. They behave like neurotransmitters in neural tissues.
- Internals are molecules that, influenced by receptors, will form networks of interactions inside the cell.

These molecules have the same basic structure (binary string, and three domains), but have different behavior that will be described below.

1) The molecule as a finite state automaton: Biological macromolecules, such as enzymes, have spatial configurations and energy levels that are condition dependent. We represent such molecular structural complexity in the hypernetwork architecture with a finite state automaton. The molecule can be in one of the following states:

- Ready: when a molecule is waiting to be activated by one of its neighbors.
- Active: when a molecule is activated by one of its neighbors, then for one time step it can activate its neighbors if there is complementarity matching.
- Inactive: after being activated, the molecule will be inactive for one time step before going back to the *ready* state.

The state transition table of the molecules are shown in Table I.

TABLE I
STATE TRANSITION TABLE FOR HYPERNETWORK MOLECULES. WITH NO INPUT, A TIME-STEP CHANGES THE STATE.

Input	Present	Next
Activation	Ready	Active
Inhibition	Ready	Inactive
-	Active	Inactive
-	Inactive	Ready
-	Ready	Ready

2) Molecular mutations: A molecule undergoes mutation under the rules described in Section III. We model mutation as the random flipping of some bits of its binary string. The molecule mutates in each of its three domains. There are two parameters that govern molecular mutations:

- The probability of molecular mutation ($MUTPROB$). Every molecule has a probability of mutation, and every molecule has the same mutation probability value.
- The percentage of bits to be randomly flipped ($PERMUT$). This percentage is to obtain the number of bits to be changed randomly in the string. The location of the bit to be changed in the string is also randomly selected. The new bit is randomly chosen, from 0,1s.

For example, lets assume we have a molecule with 14 bits in each domain (a total of 42 bits), and the parameter $MUTPROB$ is 30%, then the molecule will randomly change 4 bits from each molecular domain, and each bit has 50% chance of keeping its previous value.

B. The cellular level: intra-cellular and inter-cellular interactions

The cell is modeled by a two-dimensional cellular automaton with wrap-around. Every cell has *receptor*, *internal*, and *effector* molecules. Molecules are randomly placed in all locations of the grid. Molecules do not change locations or move, but they interact with eight neighbors as shown in Figure 4. Interactions start when a molecule is active to form excitatory or inhibitory interactions with neighbor molecules, triggering reaction cascades within the cell (forming interactions graphs).

To form excitatory interactions, the active molecule looks for all the neighbors as shown in Figure 4, if its catalytic domain matches above a threshold with the excitatory receptor

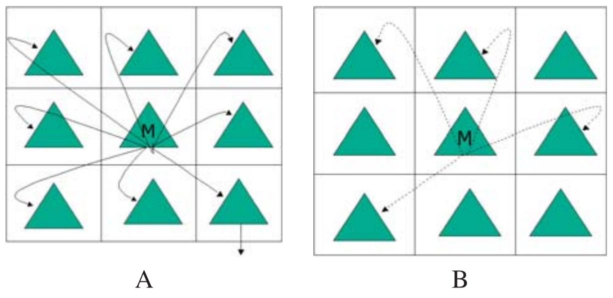


Fig. 4. (A) A molecule M, in the cell, with excitatory interactions with its neighbors. Molecule M activates its neighbor if the matching between its catalytic domain and the excitatory receptor domain of the neighbor molecule is above a threshold value. (B) A molecule M, in the cell, with inhibitory interactions with its eight neighbors. Molecule M inhibits its neighbor if the matching between its catalytic domain and the inhibitory receptor domain of the neighbor molecule is above a threshold value. The figure shows four of the eight possible inhibitions.

domain of the target molecules then the target molecules will change their state to “activated” in the next time-step (see Table I for the state transition description).

To form inhibitory interactions the active molecule looks for the neighbors as shown in Figure 4, if its catalytic domain matches above a threshold with the inhibitory receptor domain of the target molecule then the target molecule will change its state to “inactive” in the next time-step (see Table I). The target molecule will be inactive the next time-step regardless of any other activation received. Figure 4 shows a molecule M and some inhibitory interactions in a cell. The cascades of activations and inhibitions inside the cell will continue reverberating until the simulation termination condition is fullfiled.

The excitatory and inhibitory interaction molecular domains allow the formation of intra-cellular *feedback regulatory networks* that are important in learning, described in more detail in [8]. For example, in Figure 5 molecule M1 inhibits M4 at time “t+1”. The molecule is in an inactive state for one time step, then when it is back to its ready state it can be activated by another molecule. Then, M4 is activated by M2 at time “t+3”. At time “t+4” molecule M1 gets an activation but it is inhibited by M4 (feedback inhibition).

The cell to cell or inter-cellular interactions are based on their molecule interactions, which are modeled in the way neurotransmitters and receptors do in the neural tissue. The actual interactions are between an active effector molecule and the receptor molecules of target cells (see Figure 6). The interactions can be excitatory or inhibitory as described before. If the interaction is excitatory, the target molecule will became activated, triggering a cascade of molecular interactions inside the target cell. If the interaction is inhibitory, the target molecule will became inactive the next time-step, regardless of any activation from other molecules.

The cell to cell topology defines the relationships between cells. It defines the connectivity between cells, and the possible molecular interactions that can be formed between them. A cell can interact with several others at the same time. At this

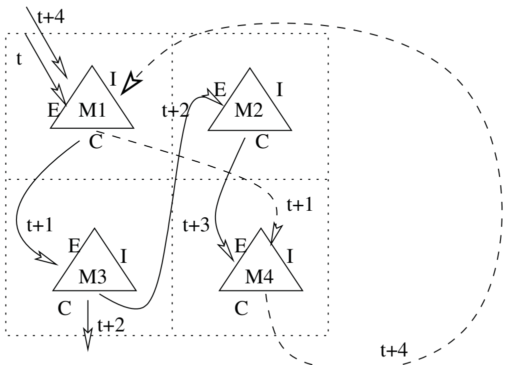


Fig. 5. Feedback regulatory networks in a cell: molecule M1 inhibits M4 at time "t+1", later after time "t+3" M4 is ready to be activated by M2. Then, at time "t+4" M4 inhibits M1. The excitatory (E), inhibitory (I), nd catalytic (C) domains of each molecule are shown.

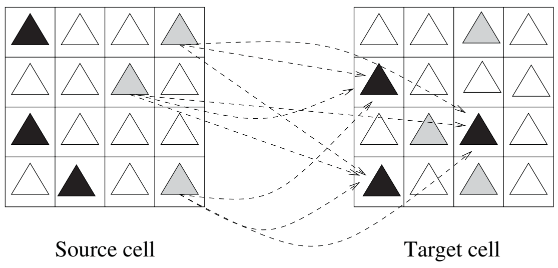


Fig. 6. Effector-receptor inter cellular interactions. The Figure shows two cells with their molecules. Receptors are black, effectors are gray, internal molecules are white. Dotted lines show the potential effector-receptor interactions between the source and target cells.

moment cells of one layer interact with the next layer, but this could change in future implementations.

C. The tissue level

A spatially organized group of cells constitutes a tissue, shown in Figure 7. The arrangement is given primarily by the cellular function. There are three cell types:

- Input cells: have receptor molecules that gather influences from the environment.
- Internal cells: do not interact with the external world.
- Output cells: have readout structures that communicate the state of the system to the environment.

At this stage of the model, the number and organization of the cells in layers is given by the designer. We programmed the possible relationships between pair of cells: the pair donor and acceptor of information. Thus, we define the cell to cell topology for information flow. Figure 7 shows the potential cell to cell interactions with double lines.

Figure 7 shows an schematics of a hypernetwork with two input cells, two internal cells in one layer, and one output cell. Input cells have receptor molecules that are activated by the binary input vector. The binary input vector is split into two bit pieces to activate at least one receptor molecule in each input cell. Once the receptor molecules in the input cells are activated, they trigger cascades of molecular reactions within the cell. Eventually the information flow will be transfered to the next layer of cells by inter-cellular interactions, and finally reach the output cells.

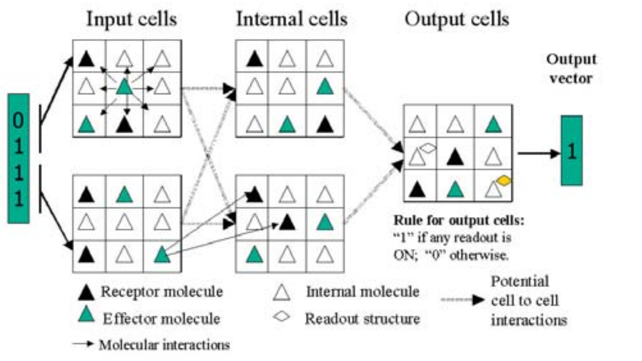


Fig. 7. Schematics of a hypernetwork with two input cells, a layers of internal cells, and an output layer with one cell. Single line arrows are molecular interactions, double lines arrows are potential cell to cell interactions.

The output cells have a few *readout structures* that read the state of the cell. The readout structures are placed randomly in the output cells, and they share locations with the molecules. If a molecule was activated at any given time, then its associated readout will be in an active state, otherwise it will be in the inactive state. The activation cascades continue to occur in the cells and across cells until any effector molecule of any output cell is activated or a long time passed and no activation occur (set to 15 time steps in the current implementation). In that case we stop the dynamics and the user can read the output of the hypernetwork by reading the state of the readout structures in the output cells. If there is any readout structure activated, then the **output of the cell** is "1", otherwise is "0". In case the output layer has more than one output cell, the output vector is formed orderly concatenating the outputs of each output cell.

An example of an actual hypernetwork is shown in Figure 8. The hypernetwork has three input cells, two layers of internal cells with three cells each, and a output cell. The input vector activates receptor molecules of the input layer, cascades of molecules interactions are formed and, in readout molecules of the output cell is activated, meaning output one.

III. The molecule evolution based learning algorithm

A hypernetwork organism is reproduced with molecular mutation (evolution at the molecular level), then its performance or fitness is evaluated. The performance of the children and parent are compared, and the best one is selected as the organism to be reproduced in the next loop or epoch (evolution at the population level).

The hypernetwork read the graph data by receptor molecules in the input cells (see Figure 7), triggering dynamical formation of networks of interactions of molecular interactions. Finally, the influences arrive at the effector molecules of output

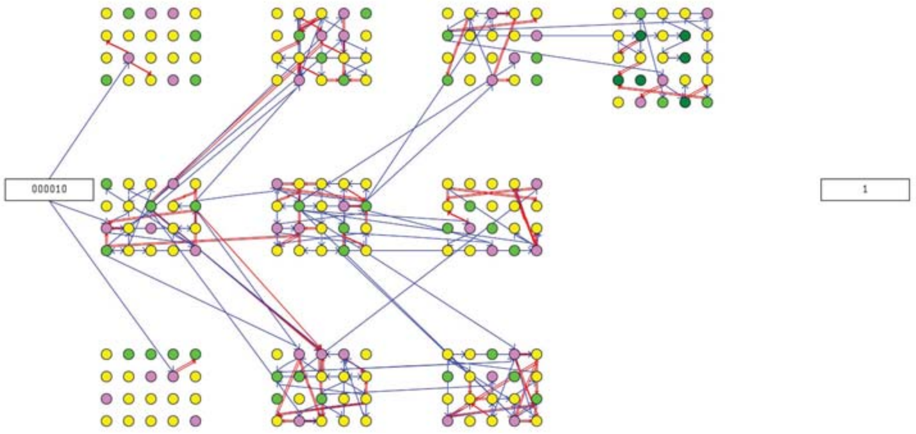


Fig. 8. A hypernetwork forming a graph in the process of information. Molecules are shown in color showing their states (receptor is lilac, effector is green, internal is yellow, readout is darkgreen). Activation edges are in blue, and inhibitions in red. In this example, for the input vector "000010", the resulting output vector is "1".

cells, where the information is gathered from the outside by means of the readout structures. The output of the organism is binary string, orderly formed by the concatenation of states from the output cells.

The output string is compared with a desired one to measure the performance of the organism for this particular input. Then, the process is applied to every input vector of the set to be learned by the algorithm. This iteration is called an *epoch*. After the algorithm has processed all the input vectors from the set, its global performance is evaluated. A detailed description follows here:

A. Initialization of hypernetwork organisms

The initial organism is created with a previously established molecular size, cellular size, possible cell to cell interactions, and number of cells. The molecular structures are initialized randomly, as are the locations of the different molecular types, and readout structures in the output cells.

B. Testing the performance of an organism

The next three steps are performed for every input vector.

- Feeding a graph into the hypernetwork.
  The input vector is split into small two bit fractions. Each of these fractions will influence an input cell (via activation of its receptors). Molecular receptors use only the first two bits of their receptor domain to match the two bit input fraction. A receptor molecule of an input cell is activated only with the matching is 100% with the input fraction. Every input cell has at least four different receptor molecules, thereby assuring that at least one of them will be activated by each fraction of the input vector. Thus, a receptor molecule is activated in every input cell by the input vector.
- Forming networks of interactions.
  Once the receptor molecules of the input cells are activated, they will activate their neighbors. In this way the influences travel through the cell until they reach effector molecules. Once the effector molecules are active, they will search for receptor molecules of other cells, according to the cell to cell topology. The cell-cell interactions are formed dynamically, based on which effectors were activated at that time.
- Generation of output vectors.
  Eventually, the cascades of interactions will activate an effector molecule of any output cell, or after a long time passed from the initial activation (set to 15 time steps). At this point, the simulation stops and the output of the cell is evaluated according to the state of the readout structures. The global state of each output cell is "1" if there is at least one readout structure activated, otherwise its global state will be "0". The output vector is formed by the concatenation of all the cellular binary states. Then the Hamming distance from the output vector (O_i) to the desired output (D_i) vector is measured (O_i - D_i)*100/k, where k is the length of the vectors.

The performance of the organism is obtained from the sum of distances evaluated previously for each input vector. Thus, for the n (Input, Output) pairs of vectors, the performance P_O is defined as follows:

$$P_0 = 1 - (\sum_{i=1}^{n}(O_i - D_i) * 100/k)/(100 * n).$$

C. Reproduction with molecular mutation

Once the performance or fitness of the organism is obtained and if it is below 100% learning or a termination condition is not fulfilled, the organism is reproduced with mutation, to generate a predetermined number of children. When generating the children, every molecule has a probability ($MUTPROB$) of mutation, independent of any circumstance. The molecule to be mutated changes according to the rules describe above. Once the children have been generated, we evaluate their performances to solve the problem. The organism with best performance is selected for the next iteration loop or epoch.

The variation - selection algorithm for hypernetwork learning is shown in Figure 9.

```
Require: A number M of organisms, Input Vectors (I), Desired output vectors (D)
1:  Initialize an organism with molecules created randomly ($O_{best}$).
2:  repeat
3:      Variation: reproduce the organism ($O_{best}$) with mutation to generate M-1 children.
4:      for Each organism o of M do
5:          for Every pair (input vector $I_i$, desired output vector $D_i$), from n vectors do
6:              Read input vector $I_i$ into input cells
7:              repeat
8:                  Propagate interactions through cells.
9:                  Form cell to cell interactions.
10:             until Effectors of output cells are activated or 15 time steps passed.
11:             Read output vector $O_i$ from the output cells
12:         end for
13:         Evaluate the performance for each organism: $P_o = 1 - (\sum_{i=1}^{n}(O_i - D_i) * 100/k)/(100 * n)$.
14:     end for
15:     Selection: obtain the best performer $O_{best}$ (the one with $\geq P_o$).
16: until ($P_{best} = 1$) or termination condition
```

Fig. 9. The variation-selection algorithm for hypernetwork learning.

IV. Classifying graphs with the hypernetwork architecture

Classification is a form of heteroassociation between patterns of input and outputs. Thus, the classifier in response to a given input pattern, should recall precisely the class of which the input pattern is member [9]. This report shows the hypernetwork as a graph classifier system.

We experiment two types of classification problems with graphs: classify graphs into classes, and classify graphs into subgraphs. The parameter setting for all the experiments with hypernetworks is given in Table: II.

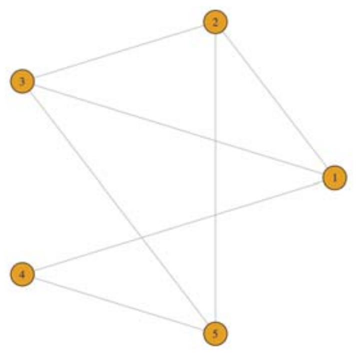
The algorithm was implemented in C++ and compiled with g++. The experiments were performed on a Intel Quad Core i7-4790 with 16GB memory PC, running Linux Ubuntu 18.04.

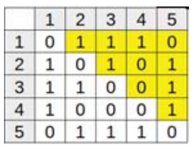
A. *Classifying five node undirected graphs into 16 classes*

The experiment uses the complete set of undirected graphs of five nodes. We construct input vector encoding the five node graphs into 10 bit input vectors (1024 or 2^{10} vectors), and the output for each vector, a class defined by the first four bits of the input vector (16 classes). For example, the graph shown in the figure 10, is encoded into the vector ”1110101011”, and belongs to the class ”1110” (the class is defined by first 4 bits of the encoded vector). Figure 13 shows two undirected graphs and their binary encoding.

TABLE II
Experimental parameter setting.

Parameter	Value
Molecule length	14 bits/domain
Threshold for excitatory matching	60%
Threshold for inhibitory matching	70%
% of molecule mutation (MUTPROB)	1
% bits to be flipped (PERMUT)	20%
% of inhibitory molecules / cell	20-30%
No. of receptor molecules	5
No. of effector molecules	5
No. of internal molecules	12-17
No. of readout structures in output cells	5
No. of individuals in the population	20





	1	2	3	4	5
1	0	1	1	1	0
2	1	0	1	0	1
3	1	1	0	0	1
4	1	0	0	0	1
5	0	1	1	1	0

A five node undirected graph ”1110101011”

Fig. 10. Example of an undirected five node graph and its adjacency matrix. The 10 bit vector is the concatenation of the upper right diagonal bits.

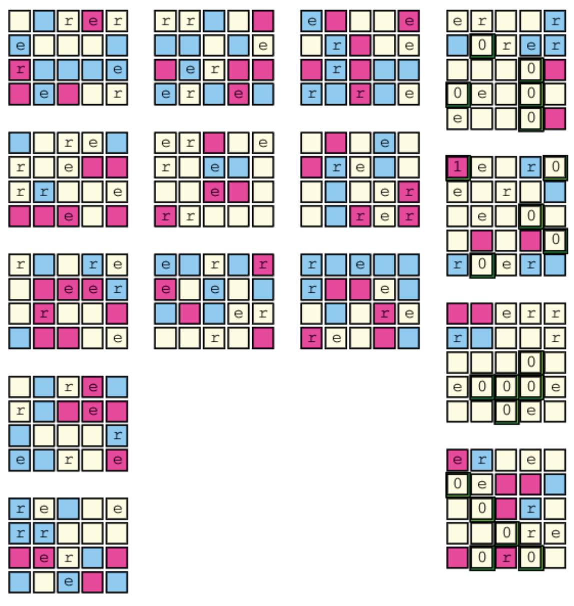


Fig. 11. Hypernetwork used to train five node graphs into 16 classes. The input layer has five cells, two internal layers of three cells each and the output layers have four cells. Labels: Effector molecules ”e”, receptor molecules ”r”. At this particular time, blue is a molecule in active state, and red is inactive. The output cells have readout molecules (”0”), if the readout activated will set the output cell to ”1”, otherwise will be ”0”. The output layer encodes (concatenates) the output vector by reading the state of each output cell.

The hypernetwork used for training is shown in Figure 11. The hypernetwork organism has five input cells, two layers of internal cells with three cells in each layer, and the output layer with four cells.

We trained the input 1024 graphs into 16 classes, and the hypernetwork learn the complete task, as shown in Figure 12.

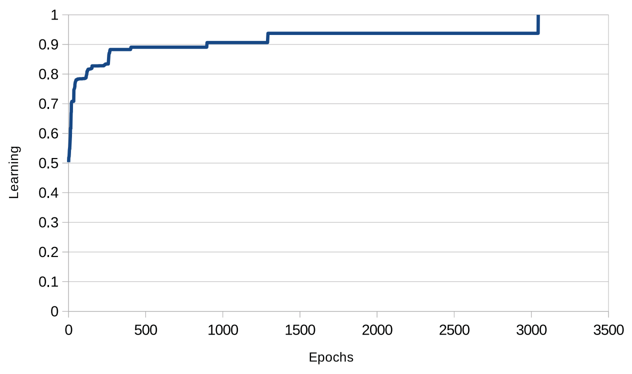


Fig. 12. Learning curve training the five node graph classification problem with 1024 input graphs. Correct classification is on the vertical axis. The hypernetwork learn to classify correctly the graphs into their classes.

B. *Classifying five node undirected graphs into four node subgraphs*

The experiment use the complete set of undirected graphs of five nodes. We construct input vectors encoding the five node graphs with 10 bit input vectors (2^{10} vectors), and the output for each vector, a class defined by a four node subgraph starting in the first node. The four node subgraph is encoded into a 6 bit vector. For example, the graphs in Figure 13 show two five node undirected graph and its four node subgraph colored in light blue.

We run the experiment twice, achieving 100% learning in one of the hypernetwoks, taking 14424 epochs in 1h 50m 4s of computer time, as shown in Figure 14.

We run 10 experiments to train a subset of graphs. The training sets were randomly chosen 80% (819 graphs), using 20% (205 vectors) as the testing graphs set. We run the experiments up to 40000 epochs. Hypernetworks achieve 100 % accuracy over the testing graph set, on 7 of the 10 runs.

C. *Classifying five node undirected graphs into three node subgraphs*

The experiment use the complete set of undirected graphs of five nodes. We construct input vectors encoding the five node graphs with 10 bit input vectors (2^{10} vectors), and the output for each vector, a class defined by a three node subgraph starting in the first node. The three node subgraph is encoded into a 3 bit vector. The graphs in Figure 15 show two examples of five node undirected graph and its three node subgraph colored in light blue.

We run the experiment twice, achieving 100 % learning with both hypernetwoks, taking less than 3000 epochs (up to 20m 35s of computer time), as shown in Figure 16.

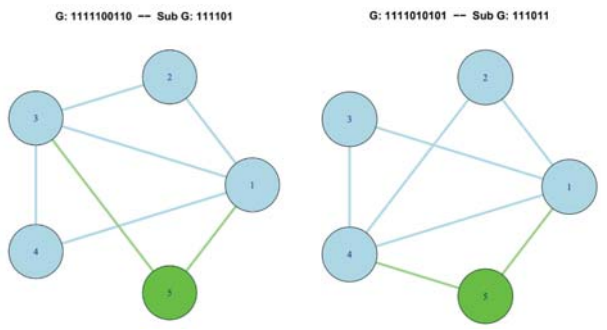

G: 1111100110 -- Sub G: 111101
G: 1111010101 -- Sub G: 111011

Fig. 13. Examples of undirected graphs with five nodes, and their corresponding four node subgraphs. Their binary representation is shown above the graph. Subgraphs are colored in light blue.

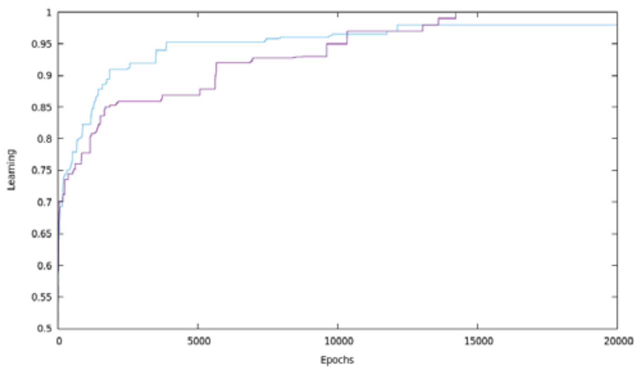


Fig. 14. Learning curves training the **five node graphs into four node subgraphs**. Correct classification is on the vertical axis. Hypernetworks learn to classify correctly the 1024 graphs.

We run 10 experiments to train a subset of graphs. The training sets were randomly chosen 80% (819 graphs), using 20% (205 vectors) as the testing graphs set. We run the experiments up to 40000 epochs. Hypernetworks achieve 100 % accuracy over the testing graph set, on 10 of the 10 runs.

V. Discussion

A. *The hypernetwork as a graph classifier system*

Results show the hypernetwork architecture performs as a graph classifier system. We achieved 100% learning to classify five node undirected graphs into 16 classes, four subgraphs, and three subgraphs. Learning subgraphs have many applications, such as locating active parts of biocompounds, vision, and other applications [10], [11].

B. *A conformation-driven evolutionary pattern recognition system*

The hypernetwork is a conformation-driven molecular pattern recognition system, where influences at molecular level percolates up to affect the behavior of the system. Evolution molded the hypernetwork behavior by adapting molecular structure and dynamics into solving the desired task (a top to bottom control mechanism). In this model, the power of the shaped-based self-assembly molecular pattern recognition capabilities of proteins and nucleic acids is captured by string

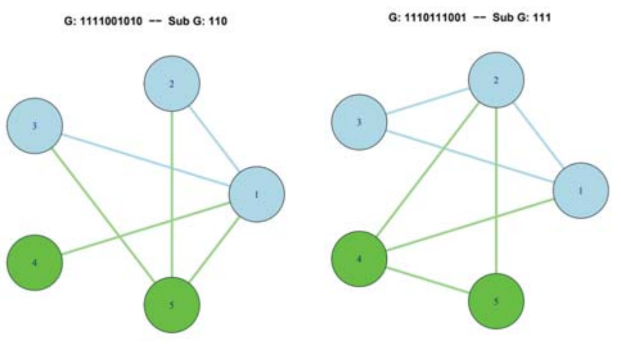


Fig. 15. Examples of undirected graphs with five nodes, and their corresponding three node subgraph. Their binary representation is shown above the graph. Subgraphs are colored in light blue.

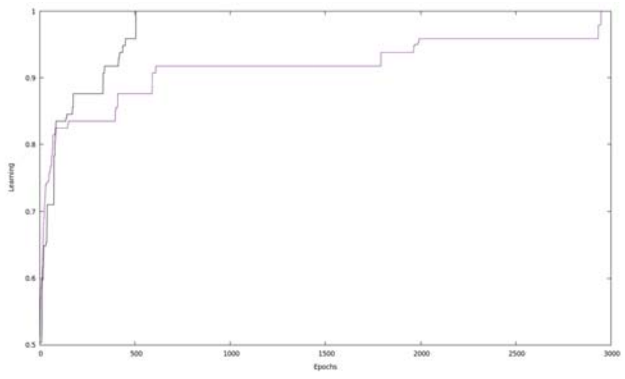


Fig. 16. Learning curves training **five node graphs into three node subgraphs**. Correct classification is on the vertical axis. Hypernetworks learn to classify correctly the 1024 graphs in less than 3000 epochs.

matching, which is used in building complex networks suitable for evolutionary learning. The molecular evolutionary algorithm is able to realize one of the many possible combinations to solve a particular task. Other approaches with evolutionary algorithms for graph classification use genetic algorithms [12], and pattern migration and competition [13].

There are major differences between neural nets and the hypernetwork architecture. Neural nets do not represent scale features, and the cell dynamics is represented by a threshold function. Neural nets store their memory in the synaptic weights. Graph Neural Networks are used to solve classification tasks [14]–[16]. For example, convolutional graph neural networks (ConvGNN), use several layers to process information: a graph convolutional layer, a pooling layer to coarsen a graph into sub-graphs (higher graph-level representations), and a readout layer to obtain a final graph representation [10]. A novel general end-to-end graph-to-sequence neural encoder-decoder model is used to map an input graph to a sequence of vectors [17].

In contrast, the hypernetwork architecture is more suitable to encode graphs on the input and ouput layers. The hypernetwork stores its memory in its molecular structures distributed across the organism, and the synaptic (inter-cellular) connections are formed dynamically, triggered by intra-cellular dynamics. The hypernetwork is molded by a variation - selection algorithm on the molecular structures, which give the capabilities of forming networks of interactions (interaction graphs), generating the output graph from the output cells.

C. Generalization performance

Preliminary experiments, in this and other studies, showed generalization performance of hypernetwork learning is comparable with other algorithms. Hypernetwork generalization is reported in [3] solving the DNA splicing problem. Gene identification in large DNA sequences involves the location of the start and stop DNA triplets (codons), exons, and introns. Typically the initial and ending sites of genes have well-defined patterns. Moreover, there are some rules for intron–exon boundaries that help to define splice sites, but this is insufficient to solve the issue completely. The problem is to discover nucleotide patterns that serve as true splice junctions in the DNA sequence over random DNA substrings. Thus, the task is to recognize exon–intron boundaries (EI or donor sites) and intron–exon boundaries (IE or acceptor sites) from large DNA sequences. This can be treated as a classification problem.

From the Genbank data set with 3186 entries, we randomly selected 5 samples of different size (100, 400, and 1000 entries each) for learning. We obtained generalization accuracy above 91% on the set with 1000 samples (see Table III). We also trained the system with a sample of 2000 entries, obtaining 0.92 accuracy on the test set of 1186 entries, outperforming backpropagation, k-NN and other algorithms on this dataset [18]. These preliminary results show hypernetwork generalization abilities, but further studies are needed to find the strengths and weaknesses of the hypernetwork architecture in addressing different problems.

TABLE III
LEARNING AND GENERALIZATION ACCURACY IN SOLVING THE DNA SPLICING PROBLEM. GENERALIZATION ACCURACY WAS OBTAINED TESTING THE EVOLVED HYPERNETWORK WITH THE TESTING SET OF 1186 ENTRIES. OBTAINED FROM [3].

Set	No. of Training instances	Generalization accuracy average
A	100	0.74
B	400	0.85
C	1000	0.91
D	2000	0.92

D. On the evolvability of the hypernetwork architecture

Biological systems have mutation buffering properties at each hierarchical level: at the molecular, cellular and organismic levels. Structural redudancies serve to buffer the effect of changes in the components, facilitating evolution [4], [19]. For example, at molecular level two versions of a protein may have the same function, but one of them may give the organism evolutionary advantages. [20] called the "bootstrap principle of evolutionary adaptability", stating that the gradualism in which a molecule function changes “is both a condition for and a consequence of evolution by variation and natural selection” [19].

The molecular buffering capabilites percolates up to the organismic level and allow the organisms to behave differently in previously unknown situations, and likewise, to produce similar behavior even when they do not have identical molecular structures in the same environment. These capabilities are the driving force on evolution, allowing mutant biologicall organisms to search for other peaks in the fitness landscape, without losing their acquired functions.

The hypernetwork architecture exhibits mutation-buffering properties that allow the system to evolve, and to maintain functionality even when some of the molecules are lost or damaged [5]. Thus, we argue that the hypernetwork could serve as a model for the physical realization of evolvable hardware.

E. The hypernetwork as a integrating framework for biologically inspired computing

The hypernetwork architecture, although independently generated, shares some structural concepts with membrane computing system [21]–[23], which is a formalism motivated by the structure and function of living cells. Elements are divided by membranes into compartments or cells, where reactions take place. The elements inside the cells evolve according to given evolution rules, and there is communication among cells. These structures are analogous to the molecules, cells, and tissue levels of the hypernetwork architecture.

The hypernetwork architecture can be extended to the population level in the following way: The algorithm reproduces copies of the best performer with offspring that have some variations at the molecular level (mutations). Several organisms, in parallel, attempt to solve the problem, improving the search in the fitness landscape. The population level is also modeled by Swarm Computing and Cultural Algorithms.

Swarm Computing [24]–[26] uses the emergent collective intelligence of agent groups to perform optimization functions. The collective behavior is a self-organizing process that emerges from the interactions of its lower-level components. The basis of the self-organizing process is the multiple interactions with both positive and negative feedbacks, and amplification of fluctuations (such as random walks) of the interactions, like the hypernetwork architecture. Thus, membrane computing and swarm computing could be considered under the more general framework of hypernetwork architecture, since it covers not only the structure, but also the learning process based on molecular evolution, finally giving the functions of the organisms.

F. Future work

- Evolution in hierarchical systems
  The current implementation of the hypernetwork architecture has molecular level evolution coupled with selection of individuals. Complexity at the molecular level could be increased with the introduction of complex approaches generating artificial molecules, such as Molnets [27]. Future implementations could include evolution at the level of cell interactions in order to better understand how cells change function in evolutionary time or in the course of development, and the role of regulation involving specialized cells.
- Experimental test-bed
  The hypernetwork architecture can be used as an experimental test-bed in the areas of artificial life, evolution and adaptability theories, machine learning in complex networks [28], and models of molecular regulation.
- Exploring the hypernetwork evolutionary learning algorithm
  The hypernetwork architecture exhibits generalization properties. However, more experimentation is needed to find the strengths and weaknesses of the hypernetwork evolutionary learning algorithm, when compared with other learning algorithms such as back-propagation and decision trees. Evolution across scales is also a potential unique feature that requires further exploration.
- Towards the physical realization of the hypernetwork
  The hypernetwork model could be implemented in Field-Programmable Devices such as FPGAs (Field-Programmable Gate Arrays) where their logic blocks could be programmed to perform matchings and thresholding functions. Currently there are many realizations of evolvable hardware, and many of them could be adapted for hypernetwork learning [29]–[31].
  Another feasible way to implement hypernetworks could be the use of single electron switch devices at the nanoscale level [32]. These devices, a realization of hybrid integrated circuits CMOL that combine CMOS devices with nanowires, can theoretically achieve densities of up to $10^{12}cm^{-2}$ [33]. The hypernetwork architecture seems a more natural model to train, from the bottom-up, the nanoelectronic neuromorphic networks (CrossNets) [34], [35].
  The molecular interaction basis of the hypernetwork, a representation of biological entities, could facilitate the design of future “bio-computers” based on networks of molecular interactions and hierarchical control. This architecture complements Conrad’s ideas of conformation-driving computing [36], where molecular interaction process form dynamic networks of interactions.

VI. Conclusions

A novel computational model of biological information processing, the hypernetwork architecture has been designed and implemented. The architecture is a theoretical model of a biological system, that can evolve new functionality. It is a multi-level vertical model that includes representations of scale, information flow, feedback regulation control, and learning. All hierarchical features are formulated in terms of interactions of elementary macromolecular subunits. The system is molded to perform classification tasks, including graph classification, through a variation-selection algorithm acting on the structure of the molecular subunits. We have shown the hypernetwork is well suited for graph classification tasks and could be implemented in novel hardware arquitectures.

Acknowledgment

The authors thank Winfried Just and Frank Stomp for their comments.

References

[1] J. Segovia-Juarez and M. Conrad, “Hypernetwork model of biological information processing,” in *Proceedings of the 1999 Congress on Evolutionary Computation*, P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, Eds., vol. 1. 6-9 July, Mayflower Hotel, Washington D.C., USA: IEEE Press, 1999, pp. 511–516.

[2] ——, “Learning with the molecular-based hypernetwork model,” in *Proceedings of the 2001 Congress on Evolutionary Computation*, J.-H. Kim, Z. Byoung-Tak, G. Fogel, and I. Kuscu, Eds., vol. 2. 27-30 May, COEX, Seoul, Korea: IEEE Press, 2001, pp. 1177–1182.

[3] J. Segovia-Juarez, S. Colombano, and D. Kirschner, “Identifying splice sites using hypernetworks with artificial molecular evolution,” *BioSystems*, vol. 87, no. 2-3, pp. 117–124, February 2007.

[4] M. Conrad, “The mutation buffering concept of biomolecular structure,” *Proc. Int. Symp. Biomol. Struct. Interactions, Suppl. J. Biosci.*, vol. 8, no. 3-4, pp. 669–679, 1985.

[5] J. Segovia-Juarez and S. Colombano, “Mutation-buffering capabilities of the hypernetwork model,” in *Proceedings of the Third NASA/DoD Workshop on Evolvable Hardware*, D. Keymeulen, A. Stoica, J. Lohn, and R. S. Zebulum, Eds. 12-14 July, Long Beach, California, USA: IEEE Computer Society, 2001, pp. 7–13.

[6] M. Conrad, “Emergent computation through self-assembly,” *Nanobiology*, vol. 2, pp. 5–30, 1993.

[7] ——, “Molecular computing: The lock-key paradigm,” *Computer*, vol. 25, no. 11, pp. 11–20, 1992.

[8] J. Segovia-Juarez and S. Colombano, “The effect of molecular inhibition on evolutionary learning: Studies in the hypernetwork architecture,” *BioSystems*, vol. 68, no. 2-3, pp. 187–198, 2003.

[9] J. M. Zurada, *Introduction to Artificial Neural Systems*. St. Paul, New York, Los Angeles: West Publishing Company, 1992.

[10] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A comprehensive survey on graph neural networks,” *CoRR*, vol. abs/1901.00596, 2019. [Online]. Available: http://arxiv.org/abs/1901.00596

[11] W. L. Hamilton, R. Ying, and J. Leskovec, “Representation learning on graphs: Methods and applications,” *CoRR*, vol. abs/1709.05584, 2017. [Online]. Available: http://arxiv.org/abs/1709.05584

[12] E. Barbu, R. Raveaux, H. Locteau, S. Adam, P. Hroux, and E. Trupin, “Graph classification using genetic algorithm and graph probing application to symbol recognition,” in *18th International Conference on Pattern Recognition (ICPR’06)*, vol. 3, 2006.

[13] N. Jin, C. Young, and W. Wang, “Gaia: Graph classification using evolutionary computation,” in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10.*, 2010, pp. 879–890.

[14] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1109/TNN.2008.2005605

[15] Y. Ma, S. Wang, C. C. Aggarwal, D. Yin, and J. Tang, “Multi-dimensional graph convolutional networks,” *CoRR*, vol. abs/1808.06099, 2018. [Online]. Available: http://arxiv.org/abs/1808.06099

[16] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A comprehensive survey on graph neural networks,” *CoRR*, vol. abs/1901.00596, 2019. [Online]. Available: http://arxiv.org/abs/1901.00596

[19] M. Conrad, *Adaptability: The Significance of Variability from Molecule to Ecosystem*. Plenum Press, New York and London, 1983.

[17] K. Xu, L. Wu, Z. Wang, Y. Feng, and V. Sheinin, “Graph2seq: Graph to sequence learning with attention-based neural networks,” *CoRR*, vol. abs/1804.00823, 2018. [Online]. Available: http://arxiv.org/abs/1804.00823

[18] D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell, *Machine learning, neural and statistical classification*. Ellis Horwood, 1994.

[20] ——, “Bootstrapping on the adaptive landscape,” *BioSystems*, vol. 11, no. 2-3, pp. 167–182, August 1979.

[21] G. Păun, *Membrane Computing. An Introduction*. Berlin: Springer-Verlag, 2002.

[22] G. Păun, “Membrane computing,” in *Fundamentals of Computation Theory*, ser. Lecture Notes in Computer Science, A. Lingas and B. Nilsson, Eds. Springer Berlin / Heidelberg, 2003, vol. 2751, pp. 177–220.

[23] G. Păun and G. Rozenberg, *The Oxford Handbook of Membrane Computing*, ser. Oxford Handbooks in Mathematics. , 2009, ch. An Introduction to and an overview of Membrane Computing, pp. 1–27.

[24] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. New York, NY, USA: Oxford University Press, Inc., 1999.

[25] C. Blum and D. Merkle, Eds., *Swarm Intelligence Introduction and Applications*, ser. Natural Computing. Springer, Berlin, 2008.

[26] B. Panigrahi, Y. Shi, and M.-H. Lim, Eds., *Handbook of Swarm Intelligence. Series: Adaptation, Learning, and Optimization*. Springer-Verlag Berlin Heidelberg, 2011, vol. 7.

[27] S. Colombano, M. H. New, A. Pohorille, J. Scargle, D. Stassinopoulos, M. Pearson, and J. Warren, “Evolutionary cell computing: From protocells to self-organized computing,” in *Proceedings of the Computational Aerosciences Workshop*. Moffett Field, CA.: NASA Ames Research Center, February 2000.

[28] T. C. Silva and L. Zhao, *Machine Learning in Complex Networks*, 1st ed. Springer Publishing Company, Incorporated, 2016.

[29] R. Salvador, A. Otero, J. Mora, E. de la Torre, T. Riesgo, and L. Sekanina, “Self-reconfigurable evolvable hardware system for adaptive image processing,” *IEEE Transactions on Computers*, vol. 62, no. 8, pp. 1481–1493, 2013.

[30] R. Dobai and L. Sekanina, “Towards evolvable systems based on the Xilinx Zynq platform,” in *2013 IEEE International Conference on Evolvable Systems (ICES)*, ser. Proceedings of the 2013 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE Computational Intelligence Society, 2013, pp. 89–95. [Online]. Available: http://www.fit.vutbr.cz/research/view_pub.php?id=10194

[31] Q. Shang, L. Chen, D. Wang, R. Tong, and P. Peng, “Evolvable hardware design of digital circuits based on adaptive genetic algorithm,” in *International Conference on Applications and Techniques in Cyber Intelligence ATCI 2019*, J. H. Abawajy, K.-K. R. Choo, R. Islam, Z. Xu, and M. Atiquzzaman, Eds. Cham: Springer International Publishing, 2020, pp. 791–800.

[32] S. Folling, O. Turel, and K. Likharev, “Single-electron latching switches as nanoscale synapses,” in *Proc. of Int. Joint Conf. on Neural Networks*, 2001, pp. 216–221.

[33] O. Turel and K. Likharev, “Crossnets: Possible neuromorphic networks based on nanoscale components,” *Int. J. of Circuit Theory and Applications*, no. 31, pp. 37–52, 2003.

[34] K. Likharev, “Crossnets: Neuromorphic hybrid CMOS/nanoelectronic networks,” *Science of Advanced Materials*, vol. 3, no. 3, pp. 322–331, June 2011.

[35] L. Ceze, J. Hasler, K. K. Likharev, J. . Seo, T. Sherwood, D. Strukov, Y. Xie, and S. Yu, “Nanoelectronic neurocomputing: Status and prospects,” in *2016 74th Annual Device Research Conference (DRC)*, June 2016, pp. 1–2.

[36] M. Conrad, “Surpassing computational limits with bioelectronic and molecular electronic technologies: Towards the multiscale computational architecture,” *Future Electronic Devices Journal*, vol. 6, no. Suppl. 2, pp. 46–60, 1995.