DNA Hypernetworks for Information Storage and Retrieval

Byoung-Tak Zhang and Joo-Kyung Kim

Biointelligence Laboratory, School of Computer Science and Engineering
Seoul National University, Seoul 151-742, Korea
{btzhang, jkkim}@bi.snu.ac.kr
http://bi.snu.ac.kr/

Abstract. Content-addressability is a fundamental feature of human memory underlying many associative information retrieval tasks. In contrast to location-based memory devices, content-addressable memories require complex interactions between memory elements, which makes conventional computation paradigms difficult. Here we present a molecular computational model of content-addressable information storage and retrieval which makes use of the massive interaction capability of DNA molecules in a reaction chamber. This model is based on the "hypernetwork" architecture which is an undirected hypergraph of weighted edges. We describe the theoretical basis of the hypernetwork model of associative memory and its realization in DNA-based computing. A molecular algorithm is derived for automatic storage of data into the hypernetwork, and its performance is examined on an image data set. In particular, we study the effect of the hyperedge cardinality and error tolerance on the associative recall performance. Our simulation results demonstrate that short DNA strands in a vast number can be effective in some pattern information processing tasks whose implementation is within reach of current DNA nanotechnology.

1 Introduction

Content-addressable memories or associative memories are storage devices which return stored contents from partial contents. These are contrasted to typical location-based storage devices where addresses are to be provided rather than contents. Content-addressable memories are useful in search intensive applications such as information retrieval, data compression, and database search [9]. It has long been known that human memory is based on content-addressing [8].

The property of massive parallelism along with associative search capability of DNA computing can be very useful in realizing content-addressable memory. This has been pointed out by several researchers in DNA computing community [1,10] and there are some experimental works going on in this line of research, for example [3] and [5]. However, there is lack of theoretical studies on developing systematic models of associative memory based on molecular computing or DNA nanotechnology.

C. Mao and T. Yokomori (Eds.): DNA12, LNCS 4287, pp. 298–307, 2006.

Here we propose a graphical model of associative memory, called hypernetworks, which is naturally implemented as a library of interacting DNA nanostructures. A hypernetwork is a weighted hypergraph, i.e., graphs having "hyperedges". Thus, a hypergraph can have edges connecting three or more vertices while in an ordinary graph the edges can connect maximum two vertices. As we shall see, the use of these hyperedges allows for additional degrees of freedom in representing memory elements in a network representation while preserving the mathematical tools provided by the graph theory.

The purpose of this paper is to introduce the hypernetwork model of associative memory from a theoretical point of view, and study its essential properties such as the tolerance of associative recall against errors in input and/or chemical reaction. We report on simulation results on pattern completion tasks where the corrupted, input images are to be reconstructed into the original or clean patterns.

The paper is organized as follows. Section 2 introduces the hypernetwork model of data storage. Section 3 presents a method for encoding the hypernetworks in DNA molecules. Section 4 describes a method for automatic storage of patterns on the hypernetwork along with its theoretical background. Section 5 shows the simulation results on the hand-written image data set. Section 6 draws conclusions.

2 The Hypernetwork Model

A hypergraph is an undirected graph G whose edges connect a non-null number of vertices [2], i.e. $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$, $E = \{E_1, E_2, ..., E_m\}$, and $E_i = \{v_{i1}, v_{i2}, ..., v_{ik}\}$. E_i is called hyperedges. Mathematically, E_i is a set and its cardinality is $k \geq 1$, i.e., the hyperedges can connect more than two vertices while in ordinary graphs the edges connect up to two vertices, i.e., $k \leq 2$. A hyperedge of cardinality k will be referred to as a k-hyperedge.

Figure 1 shows a hypergraph consisting of seven vertices $V = \{v_1, v_2, ..., v_7\}$ and five hyperedges $E = \{E_1, E_2, E_3, E_4, E_5\}$. A hypergraph can be represented as an incidence matrix. The incidence matrix of a hypergraph $G = (V, E)$ is a matrix $((a^i_j))$ with m rows that represent the hyperedges of G and n columns that represent the vertices of G, such that $a^i_j = 1$ if $v_j \in E_i$ and $a^i_j = 0$ if $v_j \notin E_i$. Each $(0, 1)$-matrix is the incidence matrix of a hypergraph if no row or column contains only zeros. Figure 1 also shows the incidence matrix corresponding to the hypergraph.

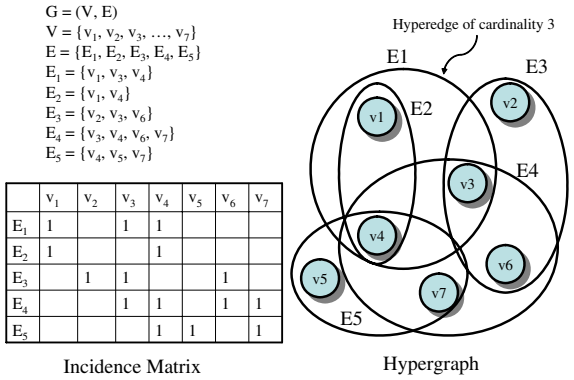
We now generalize the hypergraph into hypernetworks by assigning the weight values to the hyperedges. Formally, we define a hypernetwork as a triple $H = (V, E, W)$, where

$$V = \{v_1, v_2, ..., v_n\} \tag{1}$$

$$E = \{E_1, E_2, ..., E_m\} \tag{2}$$

$$W = \{w_1, w_2, ..., w_m\}, \tag{3}$$

where $E_i = \{v_{i1}, v_{i2}, ..., v_{im}\}$. An m-hypernetwork consists of a set V of vertices, a subset E of $V[m]$, and a set W of hyperedge weights, i.e. $H = (V, E, W)$, where



	v_1	v_2	v_3	v_4	v_5	v_6	v_7
E_1	1		1	1			
E_2	1			1			
E_3		1	1			1	
E_4			1	1		1	1
E_5				1	1		1

Fig. 1. An example hypergraph consisting of seven vertices and five hyperedges of variable cardinality. The hypernetwork can be represented a matrix, called an incidence matrix, of m rows of hyperedges and n columns of vertices.

$E = V[m]$ is a set of subsets of V whose elements have precisely m members. A hypernetwork H is said to be k-uniform if every edge E_i in E has cardinality k. A hypernetwork H is k-regular if every vertex has degree k. Note that an ordinary graph is a 2-uniform hypernetwork with $w_i = 1$.

We wish to store a data set $D = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$ in a hypernetwork so that they can be retrieved later by content. $\mathbf{x}^{(n)}$ denotes the n-th pattern to store. To do this we require the hypernetwork to represent the probabilistic distribution of the data. We define the energy of the hypernetwork

$$\begin{aligned} E(\mathbf{x}^{(n)}; W) = & -\sum_{i_1} w_{i_1}^{(1)} x_{i_1}^{(n)} - \frac{1}{2} \sum_{i_1, i_2} w_{i_1 i_2}^{(2)} x_{i_1}^{(n)} x_{i_2}^{(n)} \\ & -\frac{1}{6} \sum_{i_1, i_2, i_3} w_{i_1 i_2 i_3}^{(3)} x_{i_1}^{(n)} x_{i_2}^{(n)} x_{i_3}^{(n)} - \ldots \\ = & -\sum_{k=1}^{K} \frac{1}{k!} \sum_{i_1, i_2, \ldots, i_k} w_{i_1 i_2 \ldots i_k}^{(k)} x_{i_1}^{(n)} x_{i_2}^{(n)} \ldots x_{i_k}^{(n)}, \end{aligned} \tag{4}$$

where W represents the parameters (hyperedge weights) for the hypernetwork model. Note that $x_{i_1}^{(n)} x_{i_2}^{(n)} \ldots x_{i_k}^{(n)}$ is a combination of k components of the data item $\mathbf{x}^{(n)}$ which is represented as a k-hyperedge in the network. The probability of the data being generated from the hypernetwork is then expressed as

$$\begin{aligned} P(\mathbf{x}^{(n)}|W) &= \frac{1}{Z(W)} \exp\left[-E(\mathbf{x}^{(n)}; W)\right] \\ &= \frac{1}{Z(W)} \exp\left[\sum_{k=1}^{K} \frac{1}{k!} \sum_{i_1, i_2, \ldots, i_k} w_{i_1 i_2 \ldots i_k}^{(k)} x_{i_1}^{(n)} x_{i_2}^{(n)} \ldots x_{i_k}^{(n)}\right], \end{aligned} \tag{5}$$

where the normalizing term (known as the partition function in statistical physics) is given as

$$Z(W) = \sum_{\mathbf{x}^{(m)}} \sum_{k=1}^{K} \frac{1}{k!} \sum_{i_1, i_2, \ldots, i_k} w_{i_1 i_2 \ldots i_k}^{(k)} x_{i_1}^{(m)} x_{i_2}^{(m)} \ldots x_{i_k}^{(m)}. \quad (6)$$

In effect, the hypernetwork represents a probabilistic model of the data set using a collection of hyperedges and their weights.

3 Representing a Hypernetwork with DNA Molecules

Given two sets of data items, $D = X = \{\mathbf{x}^{(n)} | n = 1, ..., N\}$ and $Y = \{\mathbf{y}^{(n)} | n = 1, ..., N\}$, where Y is a corrupted version of X. The goal is to store X in a hypernetwork H in such a way that, given a corrupted data $\mathbf{y}^{(n)}$, the original data $\mathbf{x}^{(n)}$ is recovered or a clean version of it is reconstructed.

This task is known as pattern completion or pattern restoration for which content-addressing and associative capability is required, such as in the self-organizing systems [8,13]. The Hopfield network is another model of content-addressable memory [6]. Boltzmann machines and the Helmholtz machines [7] are generalizations of the Hopfield model by introducing the hidden variables in addition to the observable variables. All these models are based on the second-order correlations of the data. Higher-order correlations are captured by introducing hidden variables, and no explicit use of higher-order terms are made. There is, however, evidence that higher-order correlation terms are useful. The hypernetwork model has the advantage that the higher-order correlation terms can be directly represented by the hyperedges. For example, a 3-hypernetwork encodes the memory using the third-order correlation terms made of combinations of the input variables.

We now explain the method for representing the hypernetwork using DNA molecules so that the networks can be built and maintained by molecular computational operators. The idea is based on the observation that the hypernetwork is a collection of hyperedges with duplicates allowed. Basically, the original data are fragmented into a set of vertices, i.e. hyperedges, and maintained as a collection of hyperedges or, equivalently, an incidence matrix. Then the hyperedges are encoded as DNA strands. In effect, a hypernetwork is represented as a library of DNA strands where duplicates are allowed. The procedure is schematically illustrated in Figure 2.

To be more concrete, let us assume that we opted for a 3-hypernetwork model. We generate all possible hyperedges $V[3] = \{E1 = \{x_1, x_2, x_3\}, E2 = \{x_1, x_2, x_4\}, ..., E_{|V[3]|} = \{x_{n-2}, x_{n-1}, x_n\}\}$ or some subset of it to initialize the library. This results in a hypernetwork represented as a collection of hyperedges. The number of possible hyperedges increases by $2^k \times {}_nC_k$ in the number n of variables and the cardinality k of hyperedges. There should be some mechanism to choose the right hyperedges or to penalize the growth of the model complexity and we will study some of these issues in a later section.

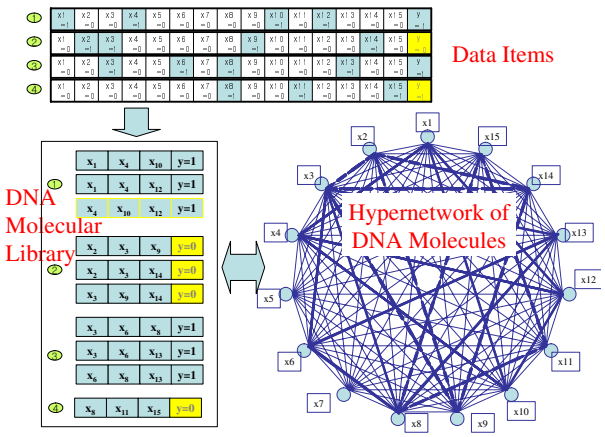


Fig. 2. General procedure for building a hypernetwork from a data set. From the data items, higher-order correlation terms are extracted and represented as hyperedges (with duplication allowed) which are then encoded as DNA strands. This library of DNA molecules represents the hypernetwork where the weights are encoded as the number of copies of the DNA molecules for hyperedges.

We use a similar method described in [12] to encode the hyperedge using a DNA strand. Each vertex (i.e. the variable and its value) in the hyperedge is encoded as a DNA sequence. For example, $x_1 = 0$ is assigned a DNA codeword 'TACAGG', where 'TACA' is for variable x_1 and 'GG' is for value '0'. In this scheme, a hyperedge $(x_1 = 0, x_3 = 1, x_4 = 0)$ is represented as 'TACAGG CTACAA GCATGG' assuming that x_3 = 'CTAC', x_4 = 'GCAT', and the value 1 is encoded as 'AA'. Then the collection of DNA-encoded hyperedges represent a hypernetwork of DNA molecules or a molecular hypernetwork.

4 Constructing a DNA Hypernetwork from Data

We now describe the procedure for building a molecular hypernetwork that fits a given data set. The basic idea is, starting with a random network, to let the network self-organize to learn the data as they are observed. The procedure is illustrated in Figure 2. The hypernetwork is represented as a collection of hyperedges, and each hyperedge is encoded as a DNA molecule, as described in the preceding section. The random k-uniform hypernetwork is then represented as a collection (or library) L of hyperedges of cardinality k where the component variables of the hyperedge and the number of copies of the hyperedges are initialized at random or according to some prior knowledge in the problem domain.

The procedure is summarized as follows:

- 1. Generate a library L of random hyperedges of cardinality k.
- 2. Get a pattern $\mathbf{x}$. Generate the hyperedges of cardinality k from $\mathbf{x}$ (with duplication permitted) into K.

- 3. (Retrieval) Find the hyperedges of L matching to those of K with error tolerance τ into M. Optionally (in case of multi-cycle retrieval), repeat this step using M as K.
- 4. (Storage) Update L by $L \leftarrow L + M + Copy(u)$ for hyperedge $u \in M$.
- 5. Go to step 2 if not terminated.

The library starts with a random collection of hyperedges of cardinality k (Step 1). As a training pattern $\mathbf{x} = (x_1, x_2, ..., x_n)$ is observed, we sample a collection K of hyperedges from $\mathbf{x}$ (Step 2). The hyperedges are generated in multiple copies where the component variables and the number of copies are chosen at random. The library L and the hyperedge collection K from the example are merged to find the matching hyperedges in L (Step 3). This can be done by hybridizing the DNA encoded hyperedges in the two collections (this to be effective, we encode the example hyperedges in complementary DNA sequences to those for the library hyperedges). The matching hyperedges are then copied by some rate, i.e $Copy(u)$ for $u \in M$, and merged with the current library L to update it (Step 4). The whole procedure is repeated for the next training pattern (Step 5). Note that the procedure makes use of relatively simple molecular operators such as selection, separation, and replication of DNA strands.

It is important to note that we allow error tolerance in this matching process where the tolerance level τ is an algorithmic parameter. The error tolerance parameter has several implications. First, it is useful to model the degree of unreliability of DNA hybridization reaction. Second, it is useful to control the generalization ability of the molecular hypernetwork memory, since the mismatches have some effect of reducing the noises in raw data. A low error tolerance (allowing only a small number of mismatches) might lead to overfitting while a high error tolerance might result in unstable learning.

It can be shown that the storage process performs gradient search to find maximum-likelihood parameters for the training data set. To see this, given a set $D = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$ of n independently and identically distributed examples, we consider the likelihood of the parameters W:

$$P(D|W) = \prod_{n=1}^{N} P(\mathbf{x}^{(n)}|W), \tag{7}$$

where W consists of the weights or the number of copies of the hyperedges of order k. Taking the logarithm of the likelihood we get

$$\begin{aligned} \ln P(D|W) &= \ln \prod_{n=1}^{N} P(\mathbf{x}^{(n)}|W) \\ &= \sum_{n=1}^{N} \left\{ \left[\sum_{k=1}^{K} \frac{1}{k!} \sum_{i_1, i_2, ..., i_k} w^{(k)}_{i_1 i_2 ... i_k} x^{(n)}_{i_1} x^{(n)}_{i_2} ... x^{(n)}_{i_k} \right] - \ln Z(W) \right\}, \end{aligned} \tag{8}$$

where Eqn. (5) is used for $P(\mathbf{x}^{(n)}|W)$. We take the derivative of the log-likelihood

$$\frac{\nabla}{\nabla w^{(k)}_{i_1,i_2,\ldots,i_k}} \ln \prod_{n=1}^{N} P(\mathbf{x}^{(n)}|W) \tag{9}$$

$$= \frac{\nabla}{\nabla w^{(k)}_{i_1,i_2,\ldots,i_k}} \sum_{n=1}^{N} \left\{ \left[\sum_{k=1}^{K} \frac{1}{k!} \sum_{i_1,i_2,\ldots,i_k} w^{(k)}_{i_1 i_2 \ldots i_k} x^{(n)}_{i_1} x^{(n)}_{i_2} \ldots x^{(n)}_{i_k} \right] - \ln Z(W) \right\}$$

$$= \sum_{n=1}^{N} \left\{ \frac{\nabla}{\nabla w^{(k)}_{i_1,i_2,\ldots,i_k}} \left[\sum_{k=1}^{K} \frac{1}{k!} \sum_{i_1,i_2,\ldots,i_k} w^{(k)}_{i_1 i_2 \ldots i_k} x^{(n)}_{i_1} x^{(n)}_{i_2} \ldots x^{(n)}_{i_k} \right] - \frac{\nabla}{\nabla w^{(k)}_{i_1,i_2,\ldots,i_k}} \ln Z(W) \right\}$$

$$= \sum_{n=1}^{N} \left\{ x^{(n)}_{i_1} x^{(n)}_{i_2} \ldots x^{(n)}_{i_k} - \langle x_{i_1} x_{i_2} \ldots x_{i_k} \rangle_{P(\mathbf{x}|W)} \right\}$$

$$= N \left\{ \langle x_{i_1} x_{i_2} \ldots x_{i_k} \rangle_{Data} - \langle x_{i_1} x_{i_2} \ldots x_{i_k} \rangle_{P(\mathbf{x}|W)} \right\}, \tag{10}$$

where the two terms in the last line are defined as

$$\langle x_{i_1} x_{i_2} \ldots x_{i_k} \rangle_{Data} = \frac{1}{N} \sum_{n=1}^{N} \left[x^{(n)}_{i_1} x^{(n)}_{i_2} \ldots x^{(n)}_{i_k} \right] \tag{11}$$

$$\langle x_{i_1} x_{i_2} \ldots x_{i_k} \rangle_{P(\mathbf{x}|W)} = \sum_{\mathbf{x}} \left[x_{i_1} x_{i_2} \ldots x_{i_k} P(\mathbf{x}|W) \right]. \tag{12}$$

The learning rule (10) suggests that maximum-likelihood is achieved by reducing the difference between the average frequencies of the hyperedges in the data set and in the hypernetwork model, as was described above. The next section studies the empirical behavior of this procedure under various experimental set-ups.

5 Simulation Results

We are interested in examining the system property of the hypernetworks in storing and retrieving patterns. In particular, we ask the following questions:

- 1. What is the effect of cardinality of hyperedges on the retrieval performance? Does increasing the cardinality help the correct retrieval or deteriorate it?
- 2. How tolerant is the hypernetwork model of associative memory against the noise or corruption of data.

To study the above questions, we used a data set consisting of handwritten numeral images. The original data came from Optical Recognition of Handwritten Digits in UCI machine learning repository[1] and we preprocessed it into a training set of 3760 examples of 8×8 bitmap. The process of image storage into the hypernetwork proceeded as described in the preceding section. To see the noise effect, another set of "corrupted" images was made by randomly toggling

[1] ftp://ftp.ics.uci.edu/pub/machine-learning-databases/optdigits/

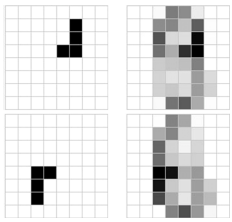


Fig. 3. Pattern completion by the hypernetwork model of associative memory. (Left column) The partial images given as input cues. (Right column) The corresponding output images reconstructed from the inputs.

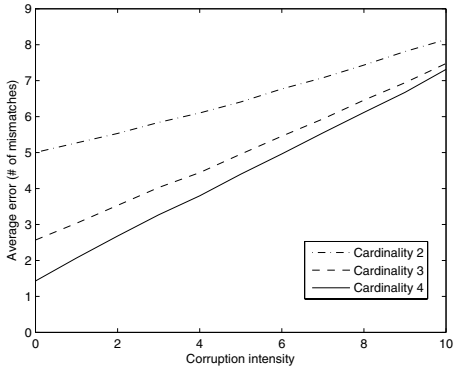


Fig. 4. The effect of cardinality k of hyperedges. Shown is the average error in image reconstruction as a function of the corruption intensity for $k = 2, 3, 4$. The hypernetworks with higher-cardinality hyperedges obtain better restoration performance than the low-cardinality networks.

the pixels of the original images. We compared the image reconstruction performances of k-uniform hypernetworks with varying $k = 2, 3, 4$, i.e., those consisting of the hyperedges of cardinality k with associated weights. We also have run the experiments by varying the level of error tolerance in matching between the hyperedges from the input image and those maintained in the library.

Figure 3 shows the images completed from the partial input. In this task, images of numeral '6' and '9' have been stored into the hypernetwork. The network recovers the appropriate patterns given partial contents as input cues. Figure 4 shows the effect of the cardinality parameter k on the associative recall of the image. The restoration error was measured as the average number of mismatches between the original image and the restored image. The results show that the restoration errors for the hypernetworks of higher k were smaller than those of lower k. It can be clearly seen that by increasing the cardinality of the hyperedges, the reconstruction error tends to decrease. This suggests the importance of higher-order correlation terms in associative recall. In this experiment, we also

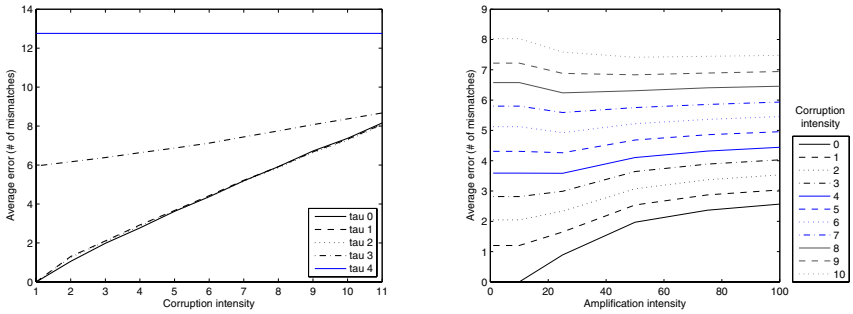


Fig. 5. The effects of the error tolerance level (left) and the amplification intensity (right). The left panel shows the stability of associative recall in the hypernetwork model against mismatches in reaction. The right panel suggests that lower amplification intensities are more appropriate to achieve a good restoration performance for slightly-corrupted data, while higher amplification intensities are not necessarily harmful for highly corrupted data.

changed the corruption levels of the images, which is depicted on the x-axis. The graph shows that the higher-order effect is especially clear when the corruption level is low.

The effect of error tolerance level τ is shown in Figure 5(left). When the number of mismatches between an input pattern and a library element is 2 or less, the library element was assumed to be matched and duplicated as if it were perfectly matched. This result shows that the performance is relatively unaffected up to some critical tolerance level (in this experiment, $\tau = 2$). However, as error tolerance increases, the average error of recovery increases because of overgeneralization. Since the error tolerance parameter indirectly reflects the unreliability in molecular reaction, this shows a stability of the hypernetwork memory in this setting of experimental parameters.

We also studied the effect of amplification intensity, i.e., the strength of learning for an observed image. The curves in Figure 5(right) show that keeping the amplification rate small helps reduce the reconstruction error, especially when the images have a low-level of corruption. However, when the images are highly corrupted, a higher rate of amplification does not necessarily hurt the performance.

6 Conclusion

We have presented a hypernetwork-based molecular architecture which allows for content-addressable storage and retrieval of patterns. The realization of this architecture using DNA molecules is described, and an algorithm is presented that automatically store data into and retrieve them from this architecture using massively parallel molecular operations. Simulation results demonstrate the possibility of using this network for pattern completion and reconstruction, i.e.

as associative memory devices. Due to lack of computing power for simulation, we were not able to perform simulations on k-hypernetworks for hyperedge cardinality $k \geq 5$. However, realized in DNA computers, we expect the molecular computational method to scale up better than in silicon computers. Another implication of the hypernetwork model is that it suggests an interesting new application of DNA-based molecular computing where a vast number of DNA molecules with short, not necessarily long, strands is useful.

Acknowledgements

This research was supported by the Ministry of Science and Technology (NRL), the Science and Engineering Foundation (Korean-German Researcher Exchange Program), and the Ministry of Industry and Commerce (MEC).

References

1. Baum, E. B., "Building an associative memory vastly larger than the brain," *Science*, 268:583-585, 1995.
2. Berge, C. *Graphs and Hypergraphs*, North-Holland Publishing, Amsterdam, 1973.
3. Chen, J. Deaton, R. and Wang, Y.-Z., "A DNA-based memory with in vitro learning and associative recall," DNA9, *LNCS* 2943:145-156, 2004.
4. Chisvin, L. and Duckworth, R. J., "Content-addressable and associative memory: Alternatives to the ubiquitous RAM," *IEEE Computer*, 22(7): 51-64, 1989.
5. Garzon, M. Bobba, K. and Neel, A., "Efficiency and reliability of semantic retrieval in DNA-based memories," DNA9, *LNCS* 2943:157-169, 2004.
6. Hopfield, J., "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci.*, 81:3088-3092, 1984.
7. Hinton, G. E. Dayan, P. Frey, B. J. and Neal, R. M. "The wake-sleep algorithm for unsupervised neural networks," *Science*, 268:1158-1161, 1995.
8. Kohonen, T. *Content-Addressable Memories*, Springer-Verlag, Berlin, 1980.
9. Paziamtzis, K. and Sheikholeslami, A., "A low-power content-addressable memory (CAM) using pipelined hierarchical search scheme," *IEEE Journal of Solid-State Circuits*, 39(9):1512-1519, 2004.
10. Reif, J.H., LaBean, T.H., Pirrung, M., Rana, V.S., Guo, B., Kingsford, C., Wickham, G.S., "Experimental construction of very large scale DNA databases with associative search capability," DNA7, *LNCS* 2340:231-247, 2002.
11. Thurber, K. J. and Wald, L. D. "Associative and parallel processors," *ACM Computing Surveys*, 7(4): 215-225, 1975.
12. Zhang, B.-T. and Jang, H.-Y., "A Bayesian algorithm for in vitro molecular evolution of pattern classifiers," DNA10, *LNCS* 3384:458-467, 2005.
13. Zhang, B.-T. Yang, J.-S., and Chi, S.-W., "Self-organizing latent lattice models for temporal gene expression profiling," *Machine Learning*, 52(1/2):67-89, 2003.