General Purpose Representation and Association Machine

Part 1: Introduction and Illustrations

Lei Wei
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816, USA
lei@ee.ucf.edu

Abstract— **Using lessons learned from error control coding, and multiple areas of life science, we propose a general purpose representation and association machine (GPRAM). GPRAM uses a versatile approach with hierarchical representation and association structures, each with different degrees of vagueness, over-completeness, and deliberate variation. GPRAM machines use vague measurements to do a quick and rough assessment on a task; then use approximated message-passing algorithms to improve assessment; and finally selects ways closer to a solution, eventually solving it. We illustrate concepts and structures using simple examples.**

Keywords-Intelligent Machine, Error Control Coding, General Purpose Systems

I. INTRODUCTION

The Human Brain is a crowning achievement of evolution, having evolved for general purposes. It can guide us on our quest to build an intelligent system that integrates various types of sensor information and processes them intelligently. Most man-made systems are built for special purposes. There are two ways to build general purpose machines. The first way (for example, modern computer) is to find one or few fixed representation and association formats, and then show that all of the other formats can be obtained by through these formats. The second way is to include as many variations as possible, to stay as broad as possible, and to move toward more specific format only when it is absolutely necessary. We shall call the first way the precise approach and the second way the versatile approach. Biological systems use the versatile approach to explore and benefit from nature without even knowing the theories behind the approach's successes. For example, bats use a form of sonar to detect obstacles.

Focusing on the fact that our brain needs ways to represent the outside world and make associations among these representations to solve general purpose problems, we aim to develop a theory outlining the functions of a general purpose representation and association machine (GPRAM) using the versatile approach.

II. LITERATURE REVIEW

The modern computer was developed in the 1940s to meet our need to handle computational tasks which our brains are not capable of performing with precision and speed. Visionary thoughts from Turing [1] and von Neumann [2] laid the foundation in term of mathematical principle and engineering structure. During the same period of time, Shannon laid the foundation of the mathematical measurement of information [3], but it does not include the *complexity*, one of key issues in our engineering activities. Even though many tasks can be done by alternative computation in Turing sense, the *complexity* advantage of some non-optimal schemes should never be ignored. Twice the difference could mean big to a bio-species. Parts of problems could be well beyond what *complexity* theory can tell us. Quantum computer may ultimately solve the *complexity* problem, but bio-systems could have already discovered many efficient and smart ways.

The key lessons learned in error control coding are listed as follows.

1. The performance averaging over randomly constructed long codes can approach the Shannon limit, so there must exist many good codes [4].
2. Many codes randomly constructed based on Tanner graphs [6] are near optimal, as long as the graphs have few small loops.
3. Information can be collected and passed around between sub-graphs at low complexity [5] [6].
4. Decoding process can be implemented iteratively [5] and it is very robust to noises and errors on the structure as well.
5. Links between iterative decoding and Pearl's belief algorithm [7], which is one of key tools to process information for Bayesian networks, have been discovered [8].
6. Many problems and solutions, including the Fourier transform and Kalman filtering problems, can be unified under the low density graph representation and iterative decoding [9].
7. Operation of code on graph and iterative decoding can be divided in three stages: repetition, random permutation, and non-linear operation, which is very similar to how neurons work [10].

Could the best known information transmission/reception mechanism help us to understand intelligent information representation and processing in a biological brain? In order to answer this question, we have spent a considerable amount of time to understand long codes [11], codes on graphs [12]-[17] and large scale random ad-hoc networks [18]. We also learn from life science, which will be summarized in Part 2. We have tried from LDPC brain, LDPC eye, to internal noisy systems [19]-[21], finally we found the foundation to describe as the GPRAM theory.

III. DEFINATION OF GPRAM

Before humans discovered scientific principles and mathematics, our ancestors used their minds to handle many tasks; for example, building homes. In the past, when peasants built their homes using mud-bricks, wood, and stone, every-effort was made to ensure the soundness of the structure. They selected and processed materials to ensure that they could estimate the stability of structure as easy as possible. Today, these rough estimations by our minds have been replaced with rigorous procedures governed by scientific principles and mathematical models. On surface, there is no reason to go back to vague estimates. However, even in a modern architectural design project, the human mind plays a key role in many aspects, including the consideration of as many factors as possible, making rough estimations and decisions, and guiding the general plan. The machine we aim to build this time is not to compute with more precision, but rather to mimic the human mind's ability to take into consideration as many broad ideas as possible and to make an educated decision as quickly as possible.

Scientists and engineers have been using intuition as a guide to their work almost every day. These intuitions are often built on their experiences, including their failures and unproven guesses. This intuition helps them make quick decisions on how to approach and tackle problems. Yet the end, they still depend on scientific methods and mathematical tools to complete the proof or perform a demonstration. It would be nice if we can build a machine to perform all these tasks, from quick judgments to definite proofs. Most of our previous AI attempts have been aimed completing this task in one machine. But the results are not very promising, largely due to complexity.

In our approach, we will split the entire process into two parts: the GPRAM part for rough and quick estimates, and the scientific and mathematical part for definite proofs. In our GPRAM machine, we will mimic part of human brain and only focus on quick and rough estimates and decisions.

Many scientists and mathematicians have been searching unsuccessfully for the precise mathematical models and coding principles of our brains. Very often, their models do not fit well with what we have observed. From our unique understanding of nature, we noticed that a machine with vague computation and approximation is sufficient to make quick estimates at a certain confidence. Once we accept the concepts of vagueness and approximation, many discrepancies disappear.

IV. DESIGN PROCEDURE

The design procedure is very different from a conventional precise approach. In a precise design, we precisely define each part of machine; then assemble the parts into a machine that performs special tasks. The design procedure for our GPRAM design is different, split into two stages: group design and individual specification. Firstly, we focus on the common features in group designs. It has three steps. (1) To stay as broad as possible from the start. (2) To determine a goal. (3) To impose rules on the GPRAM design. We call Step (3) the specification process. Every time, we add a rule, the GPRAM is narrowed down to a subset, which has a smaller degree of uncertainty. From Steps (1) to (3), we need to stay as vague as possible and cover as many tasks as possible. After that, we need to focus on special tasks or skills in the individual specification. It takes three steps. (4) To define the special tasks or skills. (5) To select a learning or training method. (6) To implement and test the GPRAM out using existing technology. This work focuses on the group design.

Suppose that we aim to find a machine design that may merge features closely matching to the human experiences, (i.e., Step 2). Now, we can search for clues in broad fields to narrow down to the architecture (i.e., Step 3). During the process, we will have learned thing well beyond scientific publication and record. At the rest steps, we can use any strategy to implement the system, as long as the used strategy does not exclude these features.

The evaluation of the success of a general purpose machines are also very different from those for machines with specific purposes, particularly during group design stage. With the versatile approach, we cannot say which individual is good or bad until we specify tasks. For precise illustration, we have to specify tasks and the particular representation. But, these specifications often mean little. So, with our GPRAM approach, we focus on common features. The key is whether the architecture can achieve these common features. We wish that the reader will also focus on these features.

In the past, simulation and computation provide evidence in verifying and optimizing a system. In a GPRAM, once we blur the boundaries between representation, association, and even structures, what can we compute or simulate? Furthermore, at any instant, the GPRAM's representations and associations are changing. Which instant should we report? All these point to that we have to use vague design (as those in steps 1 to 3), in which we will include as many features as possible. As long as we do not exclude any features during the remaining steps (Steps 4-6), we should have a valid design.

In summary, GPRAM uses a versatile approach with hierarchical representation and association structures, each with different degrees of vagueness, over-completeness, and deliberate variation. GPRAM machines use vague measurements to do a quick and rough assessment on a task; then use approximated message-passing algorithms to improve assessment; and finally selects ways closer to a solution, eventually solving it. Through these actions, GPRAM may help us deal with many computationally intensive tasks.

V. SIMPLE ILLUSTRATIONS

We use a simple example to illustrate GPRAM versatile approach. Consider two symbols A and B. A has four possible values [0, 1, 2, 3] and B has three possible values [0, 1, 2]. Let us define three tasks:

1) To indicate true (say "1") if A>1 and B>1;

2) To indicate "1" if A=0 and B=0;

3) To indicate "1" if $A \geq 1$ and B>1.

Let us see four representation cases. The first two are obtained by the precision approach. The first one has a smaller number of bits in representation and the second one is a simple and unique representation. The other two are from the versatile approach, in which we use randomly constructed and over-complete representation.

Case 1

A à x_1 x_2	B à y_1 y_2
0 à 0 0	0 à 0 0
1 à 0 1	1 à 0 1
2 à 1 0	2 à 1 0
3 à 1 1	

Case 2

A à x_1 x_2 x_3 x_4	B à y_1 y_2 y_3
0 à 1 0 0 0	0 à 1 0 0
1 à 0 1 0 0	1 à 0 1 0
2 à 0 0 1 0	2 à 0 0 1
3 à 0 0 0 1	

Case 3

A à x_1 x_2 x_3 x_4	B à y_1 y_2 y_3
0 à 0 0 1 0	0 à 0 0 0
1 à 1 0 1 1	1 à 0 1 1
2 à 1 0 0 1	2 à 1 0 1
3 à 1 1 0 1	

Case 4

A à x_1 x_2 x_3 x_4	B à y_1 y_2 y_3
0 à 0 1 0 0	0 à 0 0 0
1 à 1 0 1 1	1 à 0 0 0
2 à 1 0 1 1	2 à 1 0 1
3 à 1 0 1 1	

Case 1: Map A=[0,1,2,3] to [x_1, x_2]=[00,01,10,11], and B to [y_1, y_2]=[00,01,10]. We can perform task 1 easily as $x_1 \bullet y_1$, but become more complex for task 2 ($\overline{x_1 + x_2} \bullet \overline{y_1 + y_2}$), and task 3 ($x_1 \bullet y_1 + x_2 \bullet y_1$), where $\bullet$, $+$ and over-line denotes AND, OR and NOT logic operations, respectively. This precision approach is very efficient, only needing 4 bits. But, it is very hard to find using the versatile approach when the number of possible values becomes very large.

Case 2: Map A=[0,1,2,3] to [x_1, x_2, x_3, x_4]=[1000,0100,0010,0001], and B to [y_1, y_2, y_3]=[100, 010, 001]. We can perform task 2 with one operation ($x_1 \bullet y_1$), but two for the others, task 1 ($(x_3 + x_4) \bullet y_3$) and task 3 ($\bar{x}_1 \bullet y_3$)).

Case 3: Map A=[0,1,2,3] to [x_1, x_2, x_3, x_4]=[0010,1011,1001,1101], and B to [y_1, y_2, y_3]=[000, 011, 101]. Since x_4 duplicates x_1, it can be deleted. This is a typical. We can perform task 3 with one operation ($x_1 \bullet y_1$), but more operations for task1 ($\bar{x}_3 \bullet y_1$) and task 2 ($\bar{x}_1 \bullet \bar{y}_3$).

Case 4: Map A=[0,1,2,3] to [x_1, x_2, x_3, x_4]=[0100,1011,1011,1011], and B to [y_1, y_2, y_3] =[000, 000, 101]. This is a bad case. We do not need to evaluate it for any tasks. But, when the number of possible values becomes large, the chance of selecting a poor representation becomes much lower.

This example shows that using versatile approach, we may discover some simple solutions to perform a specific task. Complexity saving for this simple case is minor, but it could be significant for complex cases. Two key tasks in GPRAM are as follows: (1) to constantly search for simple approximation, either saving in the number of cells or saving energy or both; (2) to discover new ways.

VI. Examples with Coding Structure

In this section, we use a simple example to illustrate how GPRAM brain works. Firstly, we use the Hamming (7,4) code to illustrate some basic operations. After that, we illustrate growth, and finally extend to hierarchical structures.

A. Illustration using a (7,4) Hamming code

In Fig. 1, we draw a diagram with 2 sensors (s_1,s_2), 3 actions (a_1, a_2, a_3) and one error control code. We select Hamming (7,4) code, v_1 to v_7 denote code-word bits (called variable nodes), and C_1 to C_3, denote parity check bits (called check nodes). Edges link to nodes which indicate possible structure constraints. For example, C_1 specifies parity check constraint, i.e., $v_1 \oplus v_2 \oplus v_3 \oplus v_5 = 0$ where $\oplus$ represents XOR operation. The code generator matrix, parity check matrix, and code-word table are given in Fig. 2. The top graph of Fig. 1 is called Tanner graph which is well-known in error control coding and information theory literature [[6]]. The structure constraints are used for iterative computation of a posterior probability [[6]].

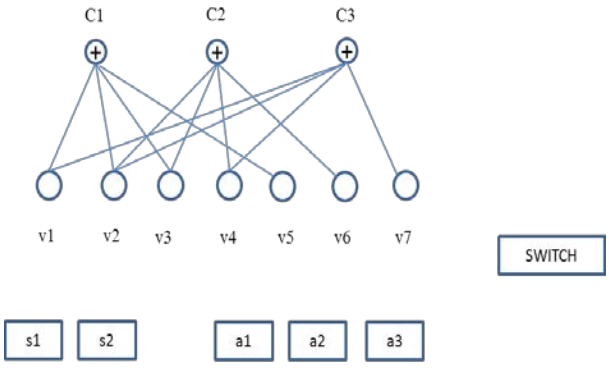

Figure 1. A simple illustration of GRPAM brain.

Let (s_1,s_2) be light sensors. Naturally, we define [s_1,s_2]=[0,0], [1,1], [1,0] for three message inputs, night, day, and seeing a human face, respectively, and actions a_1, a_2, a_3 for putting GPRAM in sleep mode, wake mode, kicking leg, respectively.

Now let see how this brain works step by step when brain connection grows initially. We just use one of many possible random selections to illustrate the case. Readers can examine others.

$$G=[I|P]=\begin{bmatrix}1&0&0&0&1&0&1\\0&1&0&0&1&1&1\\0&0&1&0&1&1&0\\0&0&0&1&0&1&1\end{bmatrix}$$

$$H=[P^t|I]=\begin{bmatrix}1&1&1&0&1&0&0\\0&1&1&1&0&1&0\\1&1&0&1&0&0&1\end{bmatrix}$$

$$\begin{bmatrix}W_0\\W_1\\W_2\\W_3\\W_4\\W_5\\W_6\\W_7\\W_8\\W_9\\W_{10}\\W_{11}\\W_{12}\\W_{13}\\W_{14}\\W_{15}\end{bmatrix}=\begin{bmatrix}0&0&0&0&0&0&0\\1&0&0&0&1&0&1\\0&1&0&0&1&1&1\\1&1&0&0&0&1&0\\0&0&1&0&1&1&0\\1&0&1&0&0&1&1\\0&1&1&0&0&0&1\\1&1&1&0&1&0&0\\0&0&0&1&0&1&1\\1&0&0&1&1&1&0\\0&1&0&1&1&0&0\\1&1&0&1&0&0&1\\0&0&1&1&1&0&1\\1&0&1&1&0&0&0\\0&1&1&1&0&1&0\\1&1&1&1&1&1&1\end{bmatrix}$$

Figure 2. Generator, parity check, and code words of Hamming code

Step 1: After experiencing a few days and nights, it starts to make connections, say s_1,s_2 to v_1 and v_2, respectively. During nights, its code will limit to those code-words with [00] at the first two bits, for example, W_8; vice versa for days, say W_{11}. ***Step 2***: Since during the night code-word W_8 is activated, v_6=1, and at the same time a_1 is activated, so both v_6 and a_1 are connected. Similarly, both v_7 and a_2 are connected. During the night, W_8, and during the day, W_{11}, are activated respectively. Quickly, we notice that W_8 also results in a_2=1. ***Step 3:*** Assume a switch box, which can switch off the sensor input during the night and switch on during the day. We will discuss the switch later. Since the day night cycle is important part of sensor input to switch box, while blocking the sensor layer will block this critical signal, we introduce the third sensor which directly connects to the switch box. ***Step 4***: During the day, it finds a new experience [s_1,s_2, v_6,v_7]=[1001], which can be only expressed by a_3. It finally finds a code word, W_1, which does not conflict with pre-arrangement, i.e., connecting a_3 to v_5. Now, it has learnt a new action, as soon as it sees a human face during the day, it will kick its leg.

Four steps are illustrated in Fig. 3 and each step is plotted in different colors for clarification.

During each trial, messages (i.e., some forms of a posterior probability) are passing between variable nodes and check nodes many times. Those unused variable nodes are free to connect other sensors or action nodes based on a simple rule: wiring together if firing together.

How to implement the switch? There are many methods: blocking signals from sensor / action layers, injecting noises into parity check units, etc. But, a simple way, which has been well-known in error control coding, is to inject heavy noises into variable nodes, then information will not be able to pass between. If this method is implemented, then an interesting feature arises, we can introduce dream-like actions to train its upper layers.

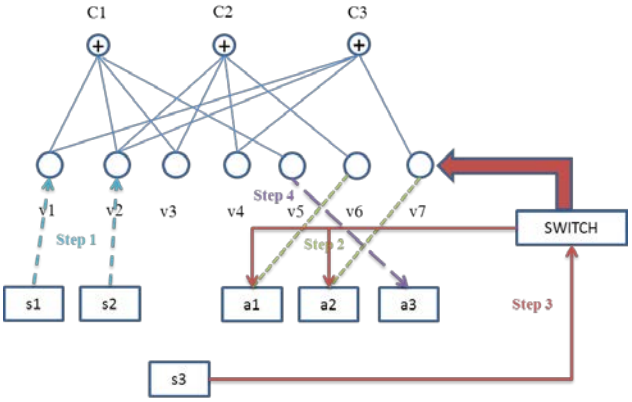

Figure 3. Final connection for a simple GPRAM system

Now, we have a product with the following actions. During the day, it will switch on to day action; unblock variable node layer. When it sees a human face, it will kick leg. During night, it will switch off variable layers and go to sleep mode.

B. Growth

Let us see one more step before we introduce hierarchical structures. Once the system can perform simple functions, it would not stop. See Fig.4. It grows more sensors (say s_1 is x_6, s_2 is x_4, and the rest of them are new sensors), variable nodes (w_1 to w_7), parity check nodes (D_1 to D_3), and action nodes. It needs to learn more functions. For simplicity, we omit actions 1 and 2, and the switch. We only focus on day activities.

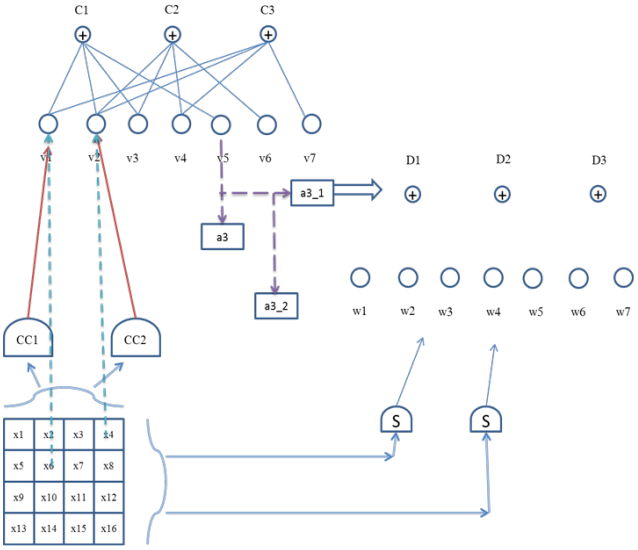

Figure 4. Growth and beyond

We introduce complex cells (CC1 and CC2) here, which connect sensors to provide global features for the coding layer to make quick guesses. Suppose CC1 connects to sensor cells (x_2, x_4, x_5, x_6, x_9, x_{12}, x_{15}), so it is for round feature; CC2 connects (x_1, x_2, x_4, x_6, x_7, x_8), so it is for horizontal patch feature. When a human face shows up, CC1

is on and CC2 is off, which match up to s_1 and s_2 descriptions. So, after many experiences, CC1 connects to v_1 and CC2 to v_2. Gradually, old connections between s_1 and v_1 and s_2 to v_2 will be replaced by these new links.

With new action units, it starts to refine its actions as well. For example, when a3 is on, it also activates a3_1 (to open a lower layer for more detail (i.e. local) feature) and at the same time activates a3_2 (to change its focus to zoom in a region, say x_6, x_7, x_{10}, x_{11}). After these actions, the visual image displayed at x_1 to x_{16} is a detailed feature of the region, and it cannot use the higher (C) layer to process, instead it uses the lower (D) layer to handle the new task. Supposing the detailed image is local feature (1) in Fig.5, combining with global feature (a), it knows this combination is a human face, not others. Gradually, it refines its kicking leg action to only this combination.

We have to keep in mind even at a lower layer, the feature is still vague, but with less degree of uncertainty. Some vagueness is a very important strategy in representation complexity saving. Once it learns one action in vagueness, it learns the action for a set of detailed things. These can explain many imaging processing features unique to human and also many psychological illusion images.

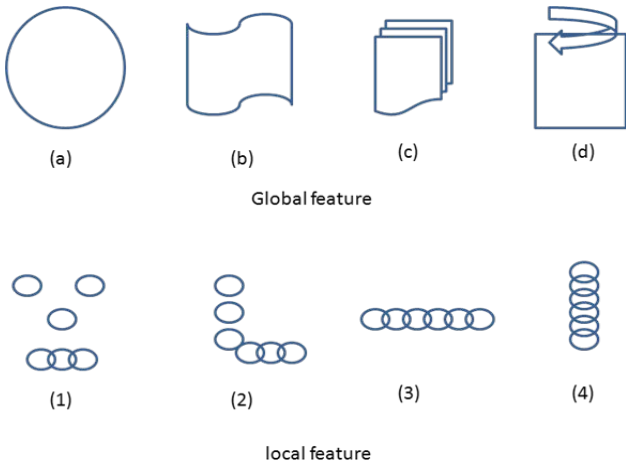

Figure 5. Local and global features

VII. Conclusions and Discussion

This is the first of three parts to introduce our GPRAM system. The second part will focus on biological implication and the third part will be coding aspect. In the second part, we will show many merging features in our machine, closely matching human experiences. These include dream-like experiences, a decrease in sleeping time as it matures, uniqueness, hierarchical efficiency, visual robustness, and specification in regions. We will also show how to explain visual illusions using our theory.

This work uses knowledge pieced together from a broad spectrum of field. Each knowledge piece is often well known within its own field, but one of our key contributions is to link them together to build a GPRAM system. Therefore, we have written this paper to be as simple as possible so a broad field of readers can comprehend it's meaning.

References

[1] A.M. Turing "On Computable Numbers, with an Application to the Entscheidungsproblem", "On Computable Numbers, with an Application to the Entscheidungsproblem: A correction", *Proceedings of the London Mathematical Society*, 2 42: 230-65. 1937, and 2 43: 544-6, 1937, respectively.

[2] von Neumann, J. , The computer and the brain, Yale University Press, USA, 2nd edition, (2000)

[3] C. E. Shannon, "A mathematical theory of communication", *Bell System Technical Journal*, (1948) 379-423, July, 623-656, October.

[4] R. G. Gallager, Information Theory and Reliable Communication. New York: Wiley, 1968.

[5] C. Berrou, A. Glavieux, and P. Thitimajshima, ``Near Shannon limit error-correcting coding and decoding: Turbo codes," *IEEE Int. Conf. Commun. (ICC)*, Geneva, Switzerland, 1993, pp. 1064-1070.

[6] R. M. Tanner, ``A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory,* Vol. 27, pp.533-547, Sept. 1981.

[7] Pearl, J. : Probabilistic Reasoning in Intelligent Systems, 2nd Ed. San Francisco, CA, USA, Kaufmann, (1988).

[8] R. J. McEliece, D. J. C. MacKay, Jung-Fu Cheng, "Turbo decoding as an instance of Pearl's ``belief propagation algorithm," *IEEE JASC* Volume: 16 Issue: 2 , Feb. 1998, Page(s): 140 -152.

[9] F. R. Kschischang, B. J. Frey and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, Vol. 47, pp.498-519, Feb. 2001.

[10] G. D. Forney, Jr., ``Codes on graphs: normal realization," *IEEE Trans. Inform. Theory*, Vol. 47, pp. 520-548, Feb. 2001.

[11] L. Wei and H.QI, ``Near Optimal Limited Search Decoding on ISI/CDMA channels and decoding of long convolutional codes," *IEEE Trans. on Inform. Theory,* Vol 46-4, July 2000, pp.1459 -1482.

[12] L. Wei, ``High Performance Iterative Viterbi Algorithm for Conventional Serial Concatenated Systems," IEEE Trans. on Information Theory, Vol. 48, July 2002, pp. 1759-1771.

[13] Q. Wang, L. Wei and R. A. Kennedy ``Iterative Viterbi Decoding, Trellis Shaping and Multilevel Structure for High-Rate Concatenated TCM," IEEE Trans on Comm., Vol. 50, Jan. 2002, pp.48-55.

[14] L. Wei, "Several Properties of Short LDPC Codes," IEEE Trans. on Communications, Vol. 52, May 2004, pp.721-728.

[15] L. Wei, "Iterative Viterbi Algorithm: Implementation Issues," IEEE Trans on Wireless Comm., Vol. 3, March 2004, pp. 382 - 386.

[16] Q. Wang* and L. Wei, "Iterative Viterbi Algorithm for Concatenated Multi-dimensional TCM," IEEE Trans. on Comm, Vol. 50, Jan. 2002, pp. 12-15.

[17] Q. Wang* and L. Wei, "Graph-Based Iterative Decoding Algorithms for Parity-Concatenated Trellis Codes," IEEE Trans. on Information Theory, Vol. 47, pp.1062-1074, March 2001.

[18] L. Wei, "Connectivity Reliability of Large Scale Random Ad Hoc Networks", Procs of MILCOM 2003, Boston, USA, October 13-16.

[19] L. Wei, "Robustness of LDPC Codes and Internal Noisy Systems", 41th Annual Allerton Conference on Communication, Control, and Computing, Allerton House, *Illinois* , USA October 2-4, 2003.

[20] L. Wei, "Biologically Inspired Statistical Matched Filter", IEEE ICC, Seoul, Korea, May 16-20, 2005.

[21] L. Wei, "Biologically Inspired Amorphous Communications", IEEE ISIT, Adelaide, Australia, Sept 3-10, 2005