Vol 17 No 10, October 2008
1674-1056/2008/17(10)/3609-07

Chinese Physics B

© 2008 Chin. Phys. Soc.
and IOP Publishing Ltd

Implementation of LT codes based on chaos*

Zhou Qian(周 茜)†, Li Liang(李 亮), Chen Zeng-Qiang(陈增强), and Zhao Jia-Xiang(赵加祥)

College of Information Technical Science, Nankai University, Tianjin 300071, *China*

(Received 2 April 2008; revised manuscript received 14 April 2008)

Fountain codes provide an efficient way to transfer information over erasure channels like the Internet. LT codes are the first codes fully realizing the digital fountain concept. They are asymptotically optimal rateless erasure codes with highly efficient encoding and decoding algorithms. In theory, for each encoding symbol of LT codes, its degree is randomly chosen according to a predetermined degree distribution, and its neighbours used to generate that encoding symbol are chosen uniformly at random. Practical implementation of LT codes usually realizes the randomness through pseudo-randomness number generator like linear congruential method. This paper applies the pseudo-randomness of chaotic sequence in the implementation of LT codes. Two Kent chaotic maps are used to determine the degree and neighbour(s) of each encoding symbol. It is shown that the implemented LT codes based on chaos perform better than the LT codes implemented by the traditional pseudo-randomness number generator.

Keywords: digital fountain, fountain codes, LT codes, Kent chaotic map
PACC: 0545

1. Introduction

Reliable transmission of data over the Internet has been the subject of many researches. In most network communication, reliability is guaranteed by the use of appropriate protocols, for example, the ubiquitous TCP/IP which ensures reliability by acknowledgements and subsequent retransmission. It is well known that the need for feedback is a bottleneck when the latency is high, or in a multicast scenario when all of the receivers in a multicast network are sending feedback to the sending server.

For these reasons, other transmission solutions have been proposed. Recently, coding theory has been applied to the transmission of large files over the Internet to avoid or reduce the need to send acknowledgements back to the sender, such as forward error correction codes. In the coding theory, the original data is encoded using some linear erasure correcting code. If during the transmission some part of the data is lost, then it is possible to recover the lost data using erasure correcting algorithms. The main benefit of this approach is that different receivers can recover different lost packets using the same erasure correcting codes. Therefore, in principle, this idea can greatly reduce the number of retransmissions.

Fountain Codes are a new class of codes originally designed for robust and scalable transmission of data over lossy computer networks. It was originally introduced in Refs.[1, 2] by Byers *et al*, based on the idealized digital fountain concept.[1] A digital fountain has properties similar to a fountain of water: when you fill your cup from the fountain, you do not care what drops of water fall in, but only want that your cup fills enough to quench your thirst. Thus data is no longer regarded as an ordered stream of bytes. For a given vector of input symbols $(x_1, \ldots, x_k)$ from the source message, a Fountain Code generates a potentially limitless stream of output symbols. Each output symbol is generated independently and randomly by sampling from a distribution D on F_2^k. The original k input symbols can be recovered with high probability from any set of n output symbols. For good Fountain codes the value of n is very close to k, and the decoding time is close to linear in k.

LT codes invented by Luby[3] were the first practical realization of the Fountain Codes. They are asymptotically optimal rateless erasure codes with

*Project supported in part by the National Natural Science Foundation of China (Grant Nos 60774088 and 10772135), the Program for New Century Excellent Talents in University of China (NCET), and the Foundation of the Application Base and Frontier Technology Research Project of Tianjin (Grant Nos 08JCZDJC21900 and 07CYBJC05800).
†E-mail: zhouqian@mail.nankai.edu.cn chenzq@nankai.edu.cn

highly efficient encoding and decoding algorithms. LT codes are rateless in the sense that the number of encoded packets that can be generated from the source message is potentially limitless. Furthermore, the encoding and decoding times are asymptotically very efficient as a function of data length. LT codes and later introduced Raptor codes which are the improvement of LT codes have attracted increasing attention in many practical applications.

In the theoretical analysis of Fountain Codes, it was assumed that for each encoding symbol, the input symbols used to generate that encoding symbol are chosen uniformly at random. However, practical implementations of LT codes cannot fully afford this randomness, because in practice the randomness is realized by a pseudo-random process, for example using a linear congruential generator of pseudo-random numbers. There has been research[4] demonstrating that in theory, such limited randomness LT codes perform almost as well as the fully random version.

Chaos is a kind of complex and random phenomenon in deterministic nonlinear dynamical systems. During recent years, there have been considerable researches about chaos systems, such as proposing new and interesting chaotic systems[5−10] and studying chaos control.[11,12] Moreover, the essential features of chaos such as sensitivity to initial conditions and control parameters, inherent randomness, the topological transitivity and ergodicity, etc., make it very appropriate to designing a pseudo-random number generator.[13,14] Many chaotic systems have been proposed as pseudo-random number generators such as Logistic map and cat map. Theoretically, chaos-based pseudo-random number generators is proved with good randomness and infinite period.

In this paper, we try to apply the inherent randomness of chaotic sequence in the implementation of LT codes. Two Kent chaotic maps are used. One is used to determine the degree of each encoding symbol, i.e. the number of input symbols which are the neighbours of an encoding symbol, according to a predetermined distribution. The other one is used to select d distinct input symbols almost uniformly at random for each output symbol. The LT codes in chaotic randomness implemented in this way are compared with the LT codes realized with the standard random number generating function in C programming language. We show that the LT codes in chaotic randomness have a better performance than the latter one.

2. Description of LT codes

2.1. Encoding process

In the theoretical construction of LT codes, each encoding symbol is generated independently of all other encoding symbols, by the following process:

(i) Randomly choose the degree d of the encoding symbol according to a predetermined degree distribution.

(ii) Choose uniformly at random d distinct input symbols as neighbours of the encoding symbol.

(iii) Set the value of the encoding symbol to be the XOR of the values of the d neighbours.

The value of each encoding symbol, together with the information about its degree and neighbours, is then put into a packet and sent to the receiver.

2.2. Decoding process

The decoder repeatedly recovers input symbols using the Decoder recovery rule in Ref.[3]. In practice, the decoding process is as follows:

(i) The decoder first receives K encoding symbols and the information about their associated degrees and sets of neighbours.

(ii) Find an encoding symbol c that has exactly one neighbour s (if there is no such encoding symbol, this decoding algorithm halts at this point, and tries to receive at least one more encoding symbol).

(a) Set $s = c$.

(b) Add s to all remaining encoding symbols r_n that also have input symbol s as a neighbour: $r'_n = r_n \oplus s$.

(c) Remove all the edges connected to the input symbol s: the degree(s) of r_n minus one, s is not the neighbour of r_n any more.

(iii) Repeat (ii) until all the input symbols are determined.

We say that the decoding process succeeds if it recovers all of the input symbols in the encoding packets. Otherwise, we say that it fails.

2.3. The degree distribution

A key to making LT codes work well is the choose a good degree distribution. Each encoding symbol has a degree chosen independently from the distribution. A good degree distribution is the Robust Soliton distribution:[3]

Definition The Robust Soliton distribution is $\mu(\cdot)$ defined as follows:

$$\rho(i) = \begin{cases} 1/k, & i = 1, \\ 1/i(i-1), & i = 2, \ldots, k, \end{cases}$$

$$\tau(i) = \begin{cases} R/(ik), & \text{for} \quad i = 1, \ldots, k/R - 1, \\ R\ln(R/\delta)/k, & \text{for} \quad i = k/R, \\ 0, & \text{for} \quad i = k/R + 1, \ldots, k, \end{cases}$$

$$\beta = \sum_{i=1}^{k} \rho(i) + \tau(i),$$

$$\mu(i) = (\rho(i) + \tau(i))/\beta, \quad \text{for} \quad i = 1, \ldots, k$$

Here k is the number of input symbols and δ is the allowable failure probability of the decoder to recover the original data for a given number K of encoding symbols, $R = c \cdot \ln(k/\delta)\sqrt{k}$ for some suitable constant $c > 0$.

The Robust Soliton distribution ensures that the expected size of the ripple[3] is large enough at each point in the decoding process so that it never disappears completely with high probability, and the decoding process will not be stopped until it succeeds. On the other hand, in order to minimize the overall number of encoding symbols used, it tries to minimize the expected ripple size so that not too many released encoding symbols redundantly cover input symbols already in the ripple.

3. The implementation of LT codes based on chaos

In this section, we introduce how we implement LT codes using the inherent randomness of chaos. In order to make LT codes work well with Robust Soliton distribution, we must use a chaotic map which is uniformly distributed. Here we choose the Kent chaotic map.

3.1. Kent map

The Kent map is one of the most studied discrete chaotic maps. It has been used to generate pseudo-random numbers in many applications such as spread spectrum communication and secure encryption. It is given by

$$x(n+1) = \begin{cases} x(n)/m, & 0 < x(n) \le m, \\ (1 - x(n))/(1 - m), & m < x(n) < 1, \end{cases}$$

where $0 < m < 1$. Hence if $x(0) \in [0, 1]$, for all $n \ge 1$, $x(n) \in [0, 1]$. The Kent map is chaotic according to the Li–Yorke definition of chaos.[15] In our application, we take $m = 0.7$.

3.1.1. Sensitive dependence on initial conditions of Kent map

Since Kent map is chaotic, it exhibits the high sensitive dependence on initial conditions of chaos. Trajectories are usually predictable for a short period of time. But over a long period, chaotic trajectories become unpredictable. Figure 1 shows the time responses from the two initial conditions of the Kent chaotic map, which are the solid line and the dotted line, respectively. In Fig.1 we see that the trajectories from the two initial conditions become very different after about 20 iterations although the difference of the two conditions being only 10^{-6}, thus the orbit of the Kent map shows a strong sensitivity to initial conditions.

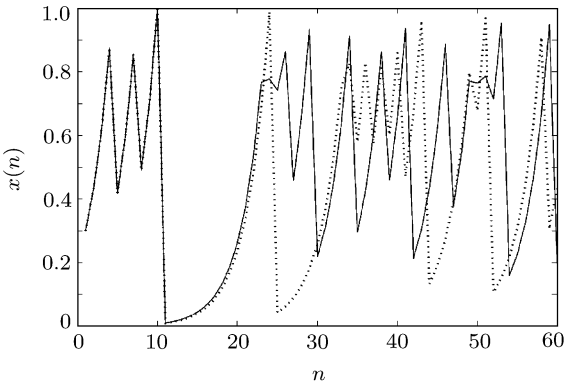


Fig.1. The trajectories of Kent map from two initial conditions with difference being only 10^{-6}.

3.1.2. Distribution characteristics of Kent map

Kent map is a chaotic map uniformly distributed in (0,1)[15] as shown in Fig.2 which is a typical time series $X(n)$ of Kent map.

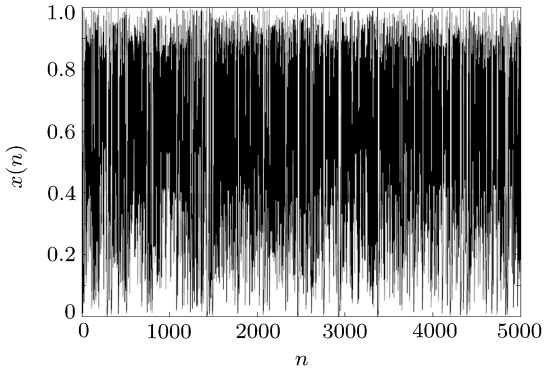


Fig.2. Time series of Kent map.

3.2. Encoding of LT codes based on Kent map

To generate an encoding packet, first we need to determine its degree according to the Robust Soliton distribution. We realize it by choosing a sample from this distribution $\mu(\cdot)$. The interval of (0,1) are divided into k non-overlapping sub-intervals according to $\mu(\cdot)$ to make each sub-interval correspond to a degree value j ($1 \leq j \leq k$, k is the total number of input symbols from the original file needed to be sent). For the ith encoding packet, we take $X(i)$ from a time series $X(n)$ of Kent map. The value of $X(i)$ will cover one of the sub-intervals and therefore determine the degree d of the ith encoding packet. Since Kent map is a chaotic map uniformly distributed in (0,1), the distribution of degrees of encoding packets conforms to the Robust Soliton distribution.

Next is to select d input symbols as the neighbours of the ith encoding packet. We take a new time series $Y(n)$ of the Kent map ($1 \leq n \leq k$). Select d largest values from the series $Y(n)$. The d input symbols with the same positions in the input vector of length k as the d largest values in the $Y(n)$ will be selected as the neighbours of the encoding packet. For example, if k=100, d=2, $Y(2)$ and $Y(79)$ are the two largest values of $Y(n)$, then the 2nd and 79th input symbols are chosen to be the neighbours of the ith encoding packet. Due to the ergodicity and inherent-randomness of chaos, each input symbol has the same probability to be chosen. In this way, we choose uniformly at random d distinct input symbols as neighbours of an encoding symbol.

Therefore, in the encoding process of LT codes based on Kent chaotic map, the ith encoding packet will be generated following the procedure shown in Fig.3(a). The procedure is repeated until all encoding packets are generated.

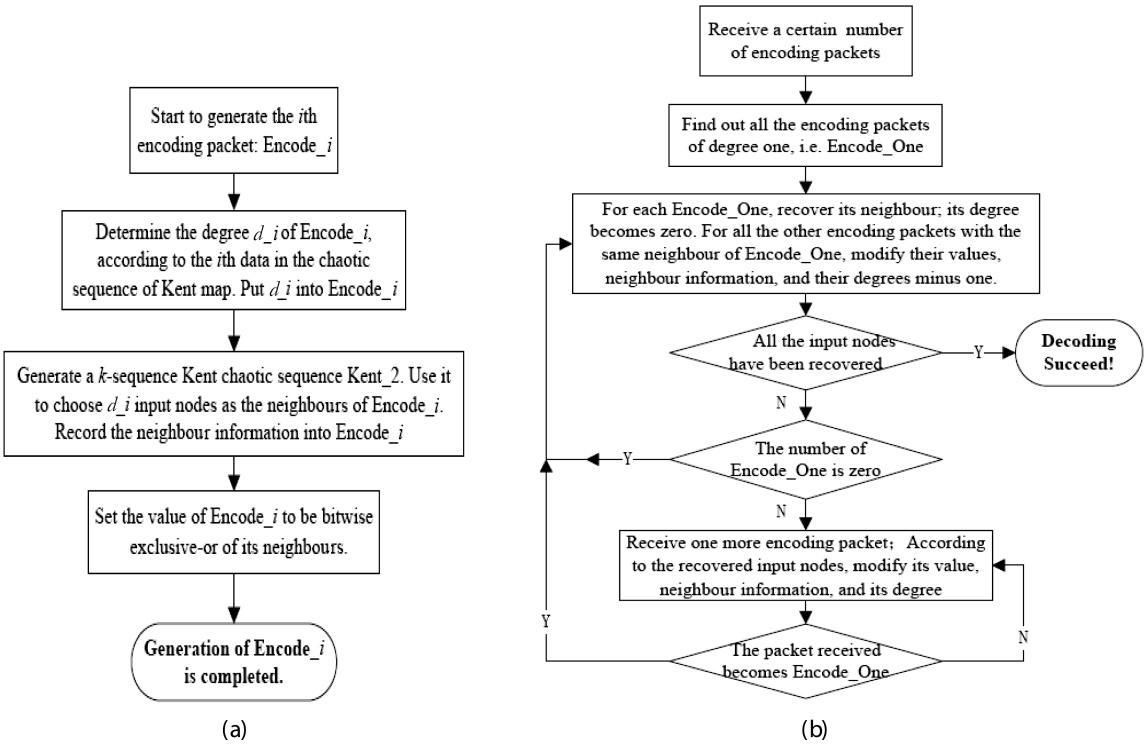


Fig.3. (a) Block diagram of encoding of LT codes based on Kent chaotic map. (b) Block diagram of decoding of LT codes based on Kent chaotic map.

3.3. Decoding of LT codes based on Kent map

The decoding process of LT codes based on Kent chaotic map is the same as the decoding process described in Sec 2.2. The block diagram of decoding of LT codes based on Kent chaotic map is shown in Fig.3(b).

4. Numerical simulations and comparison

In this section, we give our simulation results of LT codes based on chaotic map, and makes an comparison of their performance with that of LT codes implemented with the standard random number generating function rand( ) in C programming language.

For LT codes, we denote the decoding inefficiency as $1+\varepsilon$ if $(1+\varepsilon)k$ encoding packets are required to reconstruct the source file consisting of k source packets.[1] In our simulations, for original files of 1000 packets, 2000 packets and 5000 packets with the same length per packet (k=1000, 2000, or 5000), we implement LT codes with Kent map and the rand( ) function, separately. The files of k source packets are used to generate $n = 2k$ encoding packets. When $k = 1000$, we study the statistical probabilities of successful decoding of the two LT codes with the parameters $c = 0.01$, $\delta = 0.05$ over 10000 trials, respectively as shown in Fig.4. In the figure, we see that the maximum reception inefficiency of LT codes based on Kent map is 1.18, while that of LT codes based on rand( ) function is larger than 1.5, therefore the decoding efficiency of LT codes based on Kent map is much higher. Figures 5, 6 and 7 show the statistical histograms of the actual number of packets required to recover the original files of 1000, 2000 and 5000 packets with the two LT codes, separately. From the above figures, we see that the maximum number of packets required to reconstruct the source files of LT codes implemented with Kent chaotic map are much smaller than that of LT codes based on rand( ) function. In our simulations, for k=1000, 2000 and 5000, the mean decoding inefficiencies of LT codes implemented with Kent map are 1.095, 1.079 and 1.065, respectively, while those of LT codes based on rand( ) are 1.165, 1.115 and 1.087, respectively.

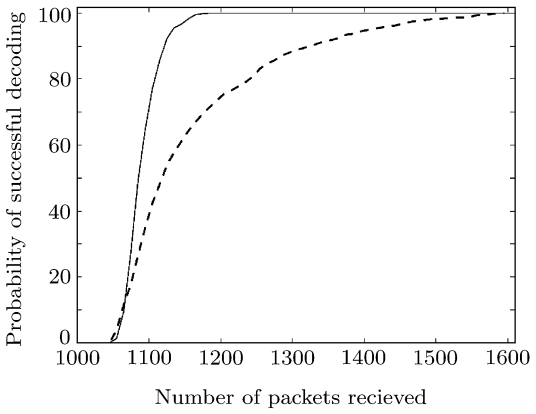


Fig.4. Probability of successful decoding of the two LT codes with the parameters k=1000, c=0.01, δ=0.05 over 10 000 trials. The solid line represents LT codes based on Kent chaotic map; the dashed line represents LT codes realized with the rand( ) function.

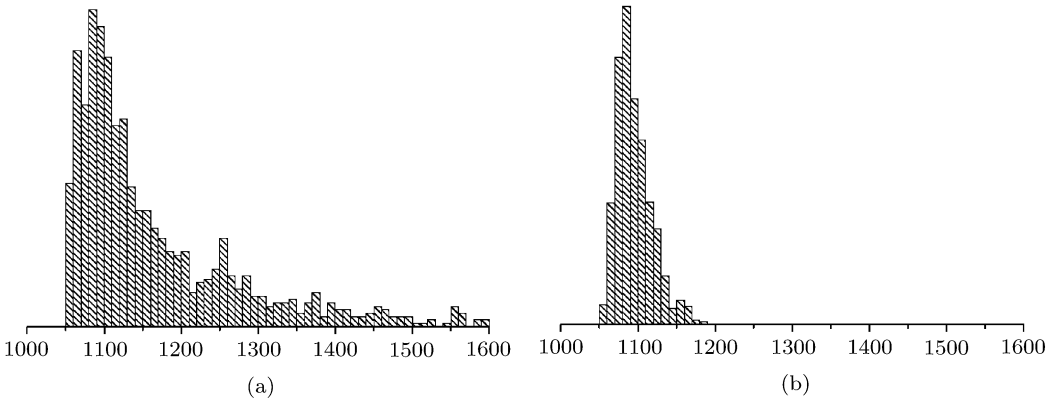


Fig.5. Histograms of the actual number of packets required to recover the original file of size k=1000 packets with the parameters c=0.01, δ=0.05. (a) LT codes based on rand( ) function; (b) LT codes based on Kent map.

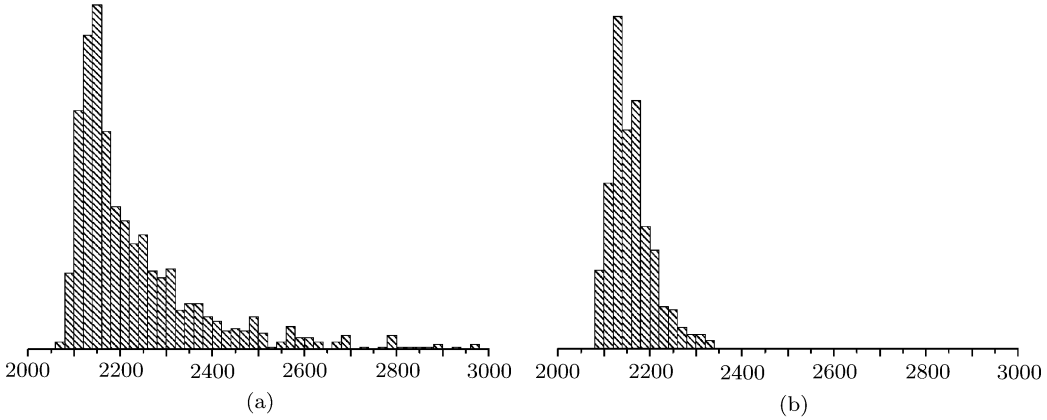


Fig.6. Histograms of the actual number of packets required to recover the original file of size k=2000 packets with the parameters c=0.013, δ=0.05. (a) LT codes based on rand( ) function; (b) LT codes based on Kent map.

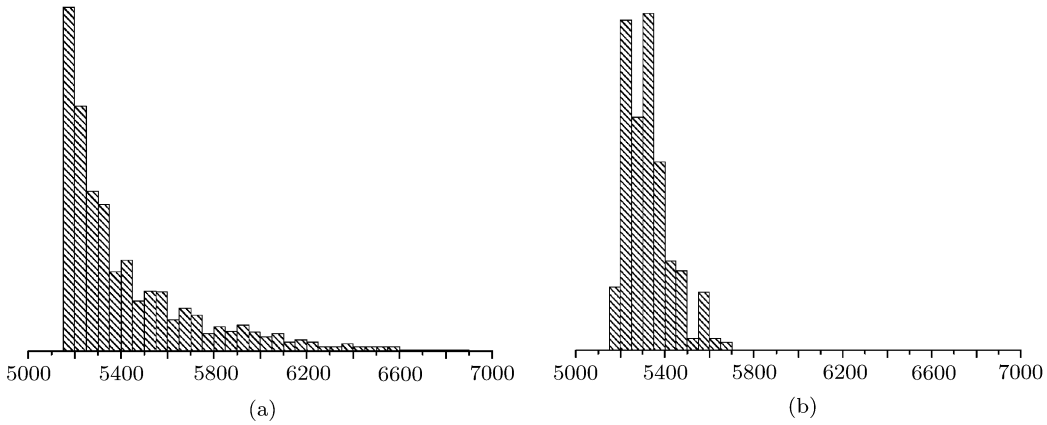


Fig.7. Histograms of the actual number of packets required to recover the original file of size k=5000 packets with the parameters c=0.01, δ=0.05. (a) LT codes based on rand( ) function; (b) LT codes based on Kent map.

Our work shows that compared to the random number generating function rand( ) in C programming language, the pseudo-randomness of the Kent chaotic map can be used to implement LT codes much closer to the full random version of LT codes. This also demonstrates the good randomness property of Kent chaotic map.

5. Conclusions

In this paper, we implement LT codes using the randomness of Kent chaotic map. In the encoding of LT codes, we use two Kent chaotic maps to determine the number of input symbols associated with each encoding symbol and select their neighbours. The performance of LT codes implemented in this way is compared with that of LT codes realized with the standard random number generating function rand( ) in C programming language. The simulation results show that the LT codes with chaotic randomness have better performances and can be tuned to recover original file with a small overhead. Therefore, compared to the random number generating function rand( ) in C programming language, the pseudo-randomness of the Kent chaotic map can be used to implement LT codes much closer to the full random version of LT code.

References

[1] Byers J W, Luby M, Mitzenmacher M and Rege A 1998 *ACM SIGCOMM Comput. Commun. Rev.* **28** 56

[2] Byers J W, Luby M and Mitzenmacher M 2002 *IEEE J. Select. Areas Commun.* **20** 1528

[3] Luby M 2002 *Proceedings of the 43rd Annual IEEE Sym-*

posium on Foundations of Computer Science p271
[4] Harrelson C, Lawrence I and Wang W 2003 *Proceedings 41st Annual Allerton Conference on Communication, Control, and Computing* **41** 492
[5] Qi G Y, Chen G R, Du S Z, Chen Z Q and Yuan Z Z 2005 *Physica* A **352** 295
[6] Qi G Y and Chen G R 2006 *Phys. Lett.* A **352** 386
[7] Qi G Y, Du S Z, Chen G R, Chen Z Q and Yuan Z Z 2005 *Chaos Solitons Fractals* **23** 1671
[8] Wang F Z, Qi G Y, Chen Z Q and Yuan Z Z 2007 *Acta Phys. Sin.* **56** 3137 (in Chinese)
[9] Wang F Z, Qi G Y, Chen Z Q, Zhang Y H and Yuan Z Z 2006 *Acta Phys. Sin.* **55** 4005 (in Chinese)
[10] Zhang Y H, Qi G Y, Liu W L and Yan Y 2006 *Acta Phys. Sin.* **55** 3307 (in Chinese)
[11] Song Y Z, Zhao G Z and Qi D L 2006 *Chin. Phys.* **15** 2266
[12] Li W L and Song Y Z 2008 *Acta Phys. Sin.* **57** 0051 (in Chinese)
[13] Mao Y B, Cao L and Liu W B 2006 *IEEE Proceedings of International Conference on Communications, Circuits and Systems* **3** 2114
[14] Falcioni M, Palatella L and Pigolotti S 2005 *Phys. Rev.* E **72** 016220
[15] Sakai H and Takumaru H 1980 *IEEE Trans. Acoustics, Speech and Signal Processing* **28** 588