LT Codes with Connection Choice

Chih-Ming Chen
Ying-ping Chen

NCLab Report No. NCL-TR-2010006
May 2010

Natural Computing Laboratory (NCLab)
Department of Computer Science
National Chiao Tung University
329 Engineering Building C
1001 Ta Hsueh Road
HsinChu City 300, TAIWAN
`http://nclab.tw/`

LT Codes with Connection Choice

Chih-Ming Chen and Ying-ping Chen
Department of Computer Science
National Chiao Tung University
HsinChu City 300, Taiwan
{ccming, ypchen}@nclab.tw

May 12, 2010

Abstract

This paper proposes *LT codes with Connection Choice* (LTCC), a modified version of LT codes, to reduce the computational cost and to enhance the applicability of LT codes. The key design of LTCC is the use of *tournament selection*, a substitute for random selection, in the encoding phase to select source symbols to process. Theoretical analyses are conducted on the missing probability of LT codes and LTCC to reveal the reason why LTCC possesses the claimed properties. Simulation results further demonstrate that LTCC can provide the same of better performance, in terms of reception overhead, with less computational cost. In summary, LTCC can be readily utilized for applications in which the number of source symbols is from hundreds to tens of thousands. For applications of more source symbols, although LTCC might not provide significantly better performance, the computational cost will still be greatly reduced.

1 Introduction

Forward Error Correction (FEC) coding is a crucial technique of error control in data transmission. In the past few years, a new class of FEC called *digital fountain codes* becomes more and more popular. The concept of fountain codes was firstly introduced in [1]. Digital fountain code works through erasure channels and corrects errors without data retransmission as one of the unique properties of FEC codes. The cost to trade such a benefit is certain extra encoding processes and redundancy to recover the source data. In the beginning of the coding process, source data are divided into pieces with an identical length. The length of each piece can be any bits or even several bytes. Afterwards, *codewords* are generated by some encoding operations and the codeword generation procedure can repeat independently and indefinitely without any limitation. Codewords are continuously created and sent out like a fountain, which is a key feature of fountain codes called *rateless.* If a receiver is interested in receiving the data, it can start to receive the data flow at any time and collect the codewords in any combination. When sufficient codewords, of which the amount is usually slightly more than that of the source data, are obtained, the source data can be fully recovered. During the transmission process, encoding information can be embedded into each codeword, and therefore, no further communication is required between sender and receiver. In summary, fountain codes are quite suitable and reliable to handle data erasure at packet level and work in the situations in which back channels are unavailable.

A simplest fountain code called *random linear fountain code* is introduced in order to help readers catch the basic idea. After an appropriate codeword length is chosen, source data can be divided into k pieces called *input symbols* and denoted as $(s_1, s_2, \ldots, s_k)$. In each independent

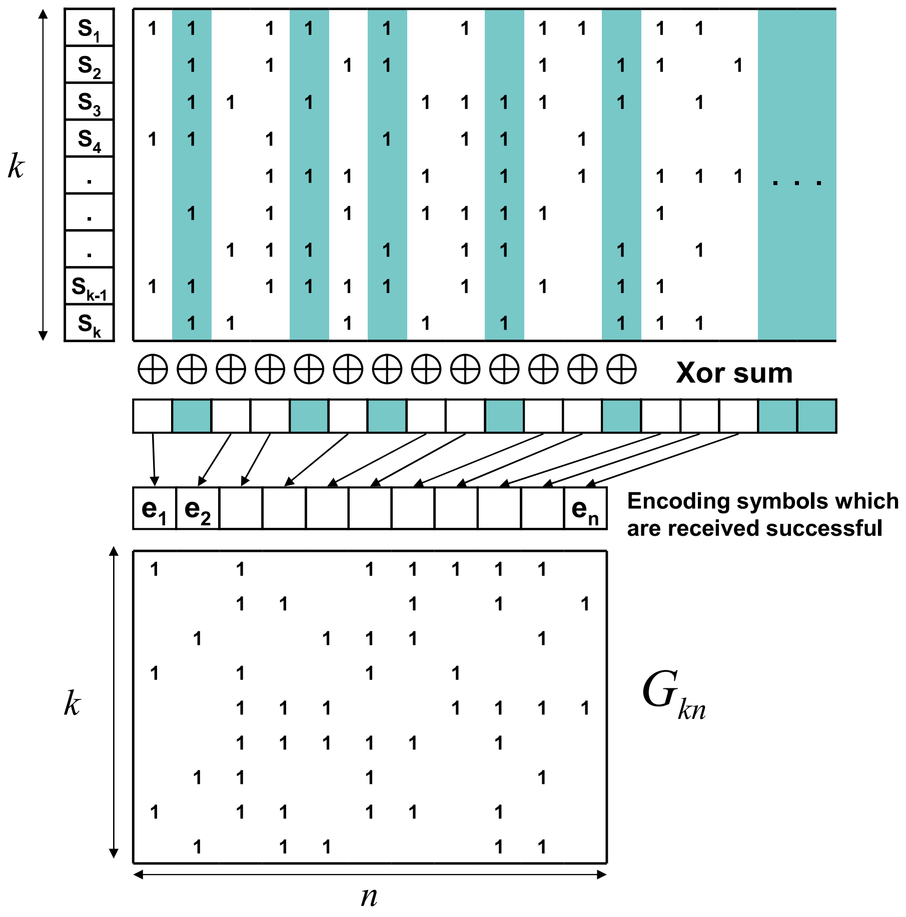


Figure 1: Random linear fountain code

run to generate a codeword, input symbols are uniformly randomly chosen and summed up with *Xor*. Figure 1 illustrates the encoding procedure in the form of matrix. Each column in the matrix is a random binary string and specifies the input symbols which are selected. Encoding symbols are generated by calculating the Xor sum of those selected input symbols in each column and sent to the receiver immediately. For example, the first column in the matrix shows that e_1 is a composition of three input symbols and can be expressed as $e_1 = s_1 \bigoplus s_4 \bigoplus s_{k-1}$. At the receiver end, some encoding symbols will arrive and others will be lost due to transmission errors. The erasure rate depends on the quality of transmission channel. In Figure 1, lost encoding symbols are colored, and received encoding symbols are denoted as $(e_1, e_2, \ldots, e_n)$. The collection of symbols e forms a $k \times n$ generator matrix G:

$$\begin{aligned} e_j &= \textstyle\sum_{i=1}^{k} s_i G_{ij} \quad ; \\ s_i &= \textstyle\sum_{j=1}^{n} e_j G_{ji}^{-1} \ . \end{aligned} \tag{1}$$

According to Equation (1), it is apparent that source data s_i can be recovered by encoding symbols e_j if inverse matrix G^{-1} exists. The first condition to ensure an invertible matrix G is to collect equal to or more than k encoding symbols. It is to say that n must be equal to or greater than k. Second, if G is invertible, there must exist at least k linear independent columns in G. [2] indicates that the probability for G to be invertible rapidly increases when slightly more than k encoding symbols are received. Suppose that $n = k + m$ in which m denotes the number of excess encoding symbols. The probability can be approximated by $1 - 2^{-m}$. In other

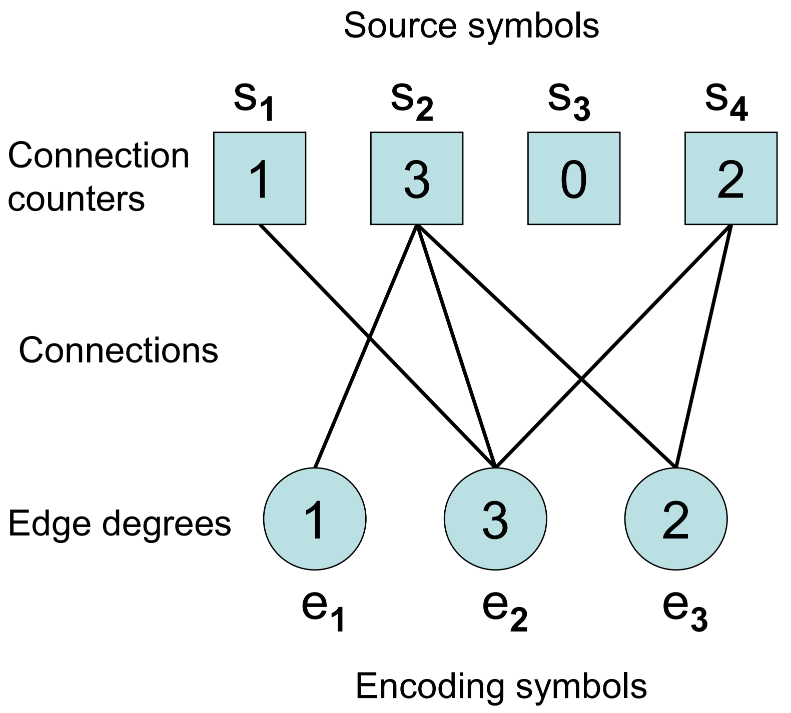


Figure 2: Bipartite structure of LT codes

words, source data can almost be reconstructed with 10 extra encoding symbols regardless of the source symbols size k. The *reception overhead* defined as $\varepsilon = m/k$ is usually used to evaluate the performance of fountain codes. Obviously, random linear fountain code is quite excellent on the indicator. However, random linear fountain code is not practical, even though it is simple and performs well in theory. In the decoding phase, a common algorithm to find the inverse matrix is Gaussian elimination and the time complexity is $\mathcal{O}(k^3)$. Huge computational cost required to reconstruct source data makes random linear fountain code impractical. As a result, both overhead and computational cost should be considered to design a good, practical digital fountain code.

Luby Transform codes (LT codes) [3] proposed by Luby in 2002 is the first practical framework and implementation of digital fountain codes. In order to reduce computational cost, *belief propagation* (BP) [4] was utilized to replace the prohibitively expensive Gaussian elimination in the decoding phase. Overhead of coding was used to trade computational cost because belief propagation is much more efficient, and slightly more encoding symbols are needed for successful decoding. The relation between input symbols and encoding symbols can be modeled as a bipartite graph, similar to the one shown in Figure 2. Since the performance of belief propagation is very sensitive to the connection structure of the graph, another key design in LT codes is the use of a probability distribution to decide the degree of each encoding symbol node in the bipartite graph. According to Luby's proposal, the performance of LT codes totally depends on the employed *degree distribution*, and hence a good probability distribution is necessary to co-operate with belief propagation. Luby also suggested the use of a family of distributions called *soliton distributions*. Mathematical verification was also provided to confirm the claimed properties of soliton distributions. More details of LT codes will be described in section 2.

Since the proposal of LT codes, quite a number of related studies have been conducted. [5] presented a method to evaluate the error probability of belief propagation applied to LT codes. [6] and [7] provided some theoretical analysis and tried to give a tighter estimation. Many studies on LT codes offered better performance by modifying the basic components. [8] and [9] attempted to find out new degree distributions which are better than soliton distributions. Some researchers aimed at the decoding procedure and proposed different mechanisms to recover source data. [10] and [11] showed similar ideas to mix belief propagation and Gaussian elimination for the balance between computational cost and overhead.

According to the related analyses, the greater k is, the better the performance of LT codes is. In fact, the mathematical verification on LT codes guarantees the performance when $k \to \infty$. For the practice purpose, Gaussian elimination becomes infeasible when k is greater than around 10^2, and LT codes start to deliver good performance when k is greater than around 10^5. However, in certain crucial networking applications, such as broadcasting and video/audio streaming services, k is right in the gap. As a consequence, improving the performance of LT codes for short data lengths or small source symbol sizes [12, 13, 8, 14, 15] becomes an important topic. To this end, the modification of LT codes proposed in the present work bridges the gap and permits the use of LT codes when k is between 10^2 and 10^5. For other attempts to enhance LT codes, [16] paid attention to the pseudo-random number generator and replaced it with chaotic sequences. In the peer-to-peer scenarios, a decomposed LT code [17] was introduced to receive encoding symbols from several senders. [18] considered source segments of different importance and prioritized encoding especially for multimedia data.

Furthermore, degree distributions dictate not only the performance but also the computational cost. The Xor-sum procedure is the largest computation consumer, because there is no need to find inverse matrices in LT codes. Hence, the number of Xor operations can roughly measure the computational cost of LT codes. Given a degree distribution $\pi(d)$, $\bar{d}$ denotes the average degree of $\pi(d)$ as

$$\bar{d} = \sum_{d=1}^{k} d \times \pi(d) \ . \tag{2}$$

Observing the bipartite graph in Figure 2, the amount of Xor operations is roughly equivalent to the number of connections between input and encoding symbols. Total connections can be obtained by multiplying the average degree $\bar{d}$ and the size of encoding symbols n. Therefore, designing a distribution with a lower average degree can directly and effectively reduce the computational cost. Nevertheless, LT codes do not take into consideration the ramifications of mean degrees. As in the encoding process of LT codes, a degree d is chosen first and then d distinct input symbols, where the strategy to select input symbols is *uniform random selection*. We can expect that each input symbol has $(\bar{d} \times n)/k$ edges on average. However, there are variances in the random process. If some input symbol is not selected as neighbor by any encoding symbols and becomes disconnected nodes, to fully recover the source data is obviously impossible. For this reason, the soliton distribution is designed to have a sufficiently large mean degree in order to allow only a very low probability for input symbols of degree zero to exist.

In addition to permitting the use of LT codes for short data lengths, the technique proposed in this paper also reduces the computational cost by lowering mean degrees while still keeping a very low probability for input symbols of degree zero to exist. An alternative input symbol selection strategy called *tournament selection* is introduced to the encoding phase and substitutes for uniform random selection. The new framework of LT codes is then named *LT codes with Connection Choice* (LTCC), because when one input symbol is needed to generate a codeword, instead of only one uniformly random selection is executed to choose a source symbol, a number of source symbols, specified with a new parameter, are selected uniformly at random to form a set of available *connection choices*. LTCC can reduce the variance of degrees effectively and be considered as a generalized version of LT codes because it is possible to configure LTCC to behave exactly as LT codes. The results of our experiments clearly show a significant performance improvement in the cases of small source symbol sizes. Not only distributions with a lower mean degree can smoothly co-operate with belief propagation but also less overhead is required to fully recover source data.

In this paper, unless defined otherwise, the notations are

- Input/source symbols size: k
- Encoding symbols size: n
- Excess symbols size: $m = n - k$
- Reception overhead: $\varepsilon = m/k$
- Average degree of distributions: $\bar{d}$
- Total connections: N

The remainder of this paper is organized as follows. Section 2 gives the detailed operations of LT codes, including the coding process and soliton distributions proposed by Luby. Section 3 introduces the new input symbol selection strategy in the encoding phase. Both the difference and influence of our mechanism will be described in detail. In section 4, numerical experiments are implemented to confirm the proposed framework, and the experimental results are presented. Finally, section 5 concludes this paper.

2 LT codes

Luby Transform codes (LT codes) is the first practical framework of digital fountain codes. Encoding and decoding in LT codes are designed to cooperate to achieve the features of fountain codes in practice, and each component of the LT code framework will be described in this section. A specific degree distribution $\pi(d)$ is adopted to decide the number of source symbols of which an encoding symbol consists. The belief propagation (BP) algorithm [4] is introduced to replace the expensive Gaussian elimination in decoding phase. Section 2.1 gives a complete introduction of the whole coding procedure. In section 2.2, two well-known degree distributions that Luby suggested for the framework are represented. According to the mathematical analysis, a basic condition, under which LT codes can complete data transmission, is discussed in section 2.3.

2.1 Encoding and decoding

Given the source data to transmit, we suppose that the source data can be cut into k input symbols, or called source symbols, of an identical length. Before each encoding symbol, or called codeword, is generated, a degree d is chosen at random according to an adopted degree distribution $\pi(d)$, where $1 \leq d \leq k$ and $\sum_{d=1}^{k} \pi(d) = 1$. The degree d decides how many distinct input symbols will be chosen to compose an encoding symbol. d input symbols, called *neighbors*, are chosen uniformly at random (u.a.r.) and accumulated by the Xor-operator. In the design of LT codes, random number generators play an essential role during the coding process. The approach employed by LT codes for sender to inform receivers of all coding information is achieved by synchronizing the random number generator with a specified random number seed.

At the receiver side, when n encoding symbols are arrived of which the number is usually slightly larger than k, belief propagation (BP) is used to recover the source data step by step. All the received encoding symbols are initially covered in the beginning. At the first step, the encoding symbols with only one neighbor can be directly released to recover their unique neighbor. When an input symbol has been recovered but not processed, it is called a *ripple* and stored in a queue. At each subsequent step, ripples are popped as a processing target one by one. A ripple is removed from the encoding symbols which have it as neighbor. If an encoding symbols has only one remaining neighbor after being removed, the releasing operation repeats and may produce new ripples to maintain a stable size of the ripple queue. Maintaining the size of the ripple queue is important for LT codes because the decoding process fails when the ripple queue is empty and covered input symbols remain. In other words, more encoding symbols are required in the decoding process. Ideally, the process succeeds if all the input symbols are recovered at the end of the decoding process.

2.2 Soliton distribution

According to the design, the behavior LT codes is completely determined by the adopted degree distribution, $\pi(d)$, and the number of received encoding symbols, n. The overhead $\varepsilon = (n-k)/k$ denotes the performance of LT codes, and ε depends on $\pi(d)$. Based on his theoretical analysis, Luby proposed the ideal soliton distribution by using which the overhead is 0, the best performance, in the ideal case.
Ideal soliton distribution $\rho(d)$:

$$\rho(d) = \begin{cases} \frac{1}{k} & \text{for} \quad d = 1 \\ \frac{1}{d(d-1)} & \text{for} \quad d = 2, 3, \ldots, k \end{cases} . \tag{3}$$

Ideal soliton distribution guarantees that all the release probabilities are identical to $1/k$ at each subsequent step. Hence, there is *one* expected ripple generated at each processing step when the encoding symbol size $n = k$. After k processing step, the source data can be ideally recovered. Fig. 3(a) shows an example of ideal soliton distribution for $k = 30$.

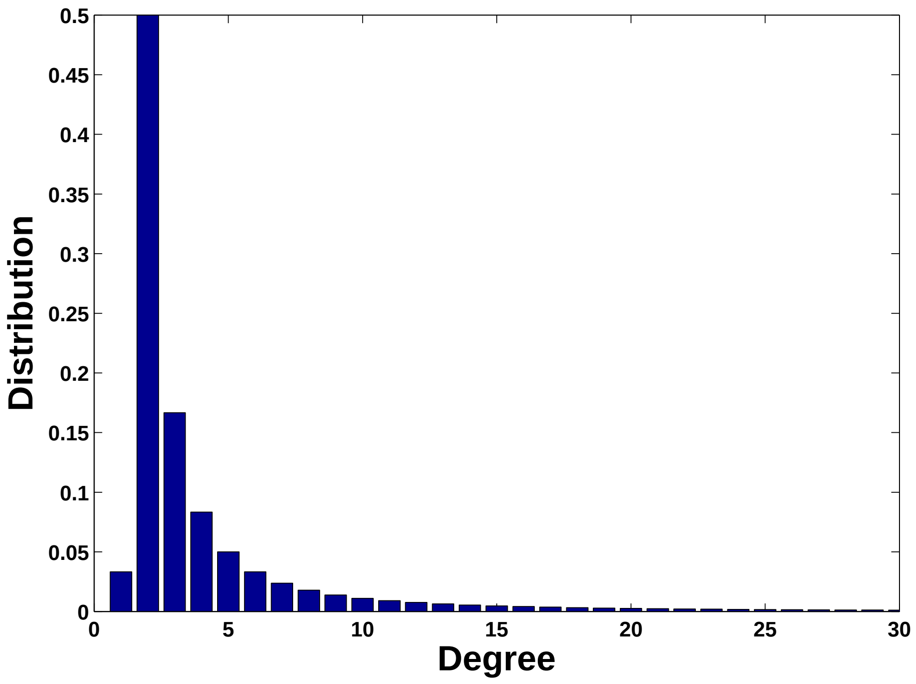
However, the ideal soliton distribution works poorly in practice. The operation of belief propagation may be suspended by a small variance of the stochastic decoding situation in which no ripple exists, because the expected ripple size is only one at any moment. According to the theory of random walk, the probability with which a random walk of length k deviates from its mean by more than $\ln(k/\delta)\sqrt{k}$ is at most δ. We can consider it as a baseline of ripple sizes which must be maintained to complete decoding. Hence, in the same paper by Luby, a modified version called *robust soliton distribution*, $\mu(d)$, was also proposed.
Robust soliton distribution $\mu(d)$:

$$R = c \cdot \ln(k/\delta)\sqrt{k} .$$

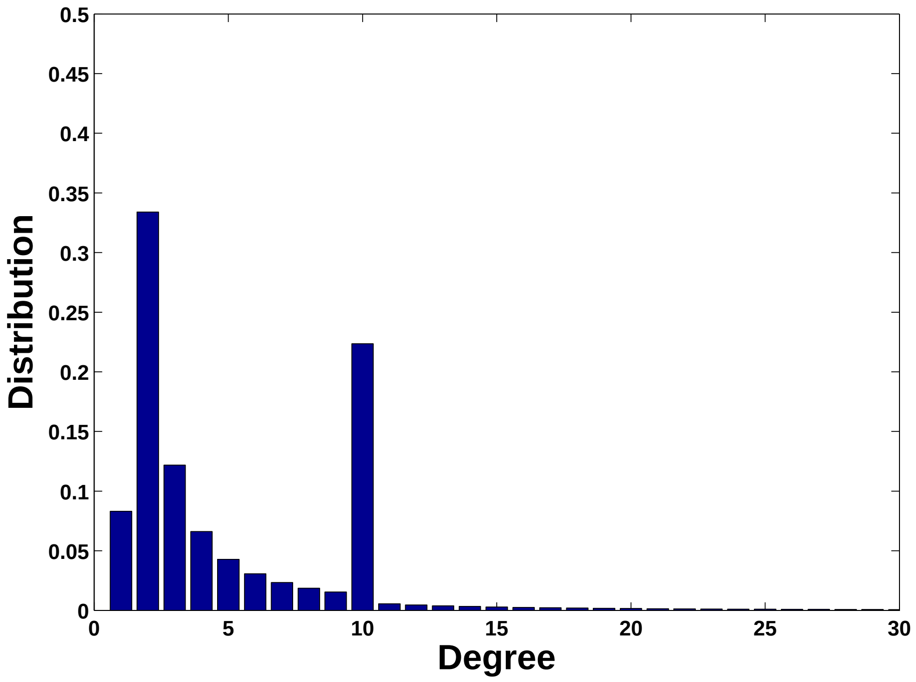
$$\tau(d) = \begin{cases} R/ik & \text{for} \quad d = 1, \ldots, k/R - 1 \\ R\ln(R/\delta)/k & \text{for} \quad d = k/R \\ 0 & \text{for} \quad d = k/R + 1, \ldots, k \end{cases} . \tag{4}$$

$$\begin{aligned} \beta &= \sum_{d=1}^{k} (\rho(d) + \tau(d)) . \\ \mu(d) &= \frac{\rho(d) + \tau(d)}{\beta} \text{ for } d = 1, \ldots, k . \end{aligned} \tag{5}$$

c and δ are parameters for tuning robust soliton distribution. c controls the mean of the degree. Small value of c increases the probability of low degrees and a large one decreases it. δ estimates that there are $\ln(k/\delta)\sqrt{k}$ expected ripple size as previously described. Fig. 3(b) gives an example of robust soliton distribution with $c = 0.1$ and $\delta = 0.1$. Robust soliton distribution can ensure



(a) Ideal soliton distribution



(b) Robust soliton distribution

Figure 3: Examples of soliton distributions ($k = 30$)

that only $n = k + \mathcal{O}(\ln^2(k/\delta)\sqrt{k})$ encodeing symbols are required to recover the source data with a successful probability at least 1-δ. Robust soliton distribution is not only viable but also practical. The analysis of robust soliton distribution based on probability and statistics is sound, especially when $k \to \infty$.

2.3 Missing probability of LT codes

In the framework of LT codes, each encoding symbol is a combination of input symbols which are selected u.a.r. It is apparently very important to make sure that all the input symbols are selected at least once. Otherwise, such *missing input symbols* which are never chosen as connection neighbors will represent the lack of information and lead to the impossibility to fully recover the source data. This situation could be easily imagined as a "ball-bin" problem: How many bins are empty in expectation if total N balls are thrown into k bins u.a.r? The probability that one particular bin is empty can be expressed as

$$(1 - \frac{1}{k})^N \backsimeq e^{-N/k} \ . \tag{6}$$

If $N = k$, the probability of a particular bin to be empty is roughly $1/e \approx 0.368$. There will be an empty one among any three bins on average, and obviously the ratio is too high to make all bins have at least one ball. More balls are needed to make sure there is no empty bin. Suppose that $1 - \delta$ denotes the probability that none of the bins is empty. The number of balls required to guarantee δ is $k \ln(k/\delta)$. Given the encoding symbol size $N > k \ln(k/\delta)$, the probability that a particular bin is empty is at most δ/k according to Equation (7).

$$e^{-N/k} < e^{-[k \ln(k/\delta)]/k} = \frac{\delta}{k} \ . \tag{7}$$

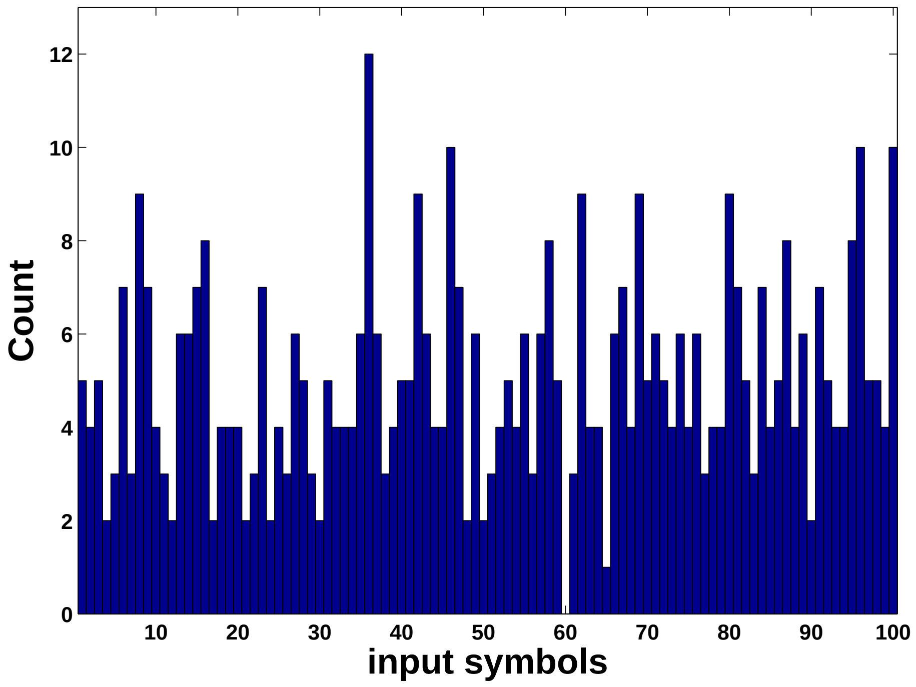
Whether or not a bin is empty is an independent event, and the probability that all the bins have balls can be simply calculated with Equation (8), in which, small δ well supports the soundness of the approximation.

$$\begin{aligned} (1 - \frac{\delta}{k})^k &= (1 - \frac{1}{k/\delta})^k \\ &\backsimeq e^{\frac{-k}{k/\delta}} \\ &= e^{-\delta} \\ &\backsimeq 1 - \delta \ . \end{aligned} \tag{8}$$

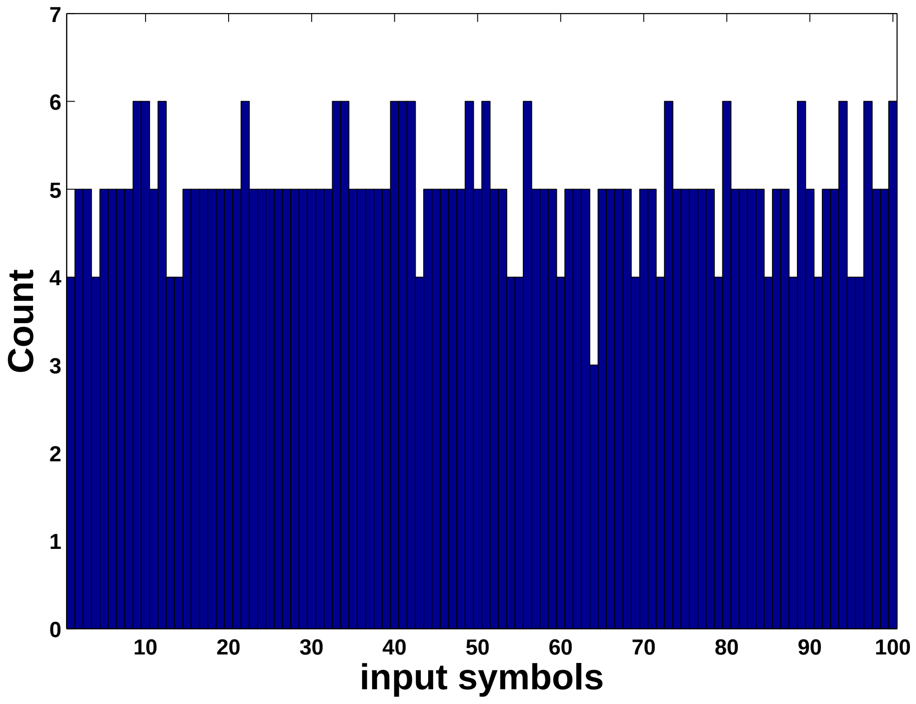
The ball-bin problem helps us to understand the situation in which input symbols are chosen. Using an insufficient amount of balls tend to leave some bins empty. Fig. 4(a) shows a histogram of 500 random choices against 100 input symbols. In this example, there is a missing input symbol which leads to the failure of decoding. Hence, roughly $k \ln(k/\delta)$ connections are required to guarantee the probability $1 - \delta$ that all the input symbols are connected at least once. Moreover, the total number of connections is decided by the average degree of the distribution and the encoding symbol size n. n is always asked to get close to k, and therefore, a viable degree distribution should be with the average degree at least $\ln(k/\delta)$:

$$\bar{d} = \frac{k \ln(k/\delta)}{n} \backsimeq \ln(k/\delta) \ . \tag{9}$$

Ideal and robust soliton distributions are both designed according to the aforementioned guideline. [3] gives the detailed derivations of the mean degree of soliton distributions. The mean



(a) Random selection (Equivalent to tournament selection with $T = 1$)



(b) Tournament selection with $T = 3$

Figure 4: Histograms of connection choice

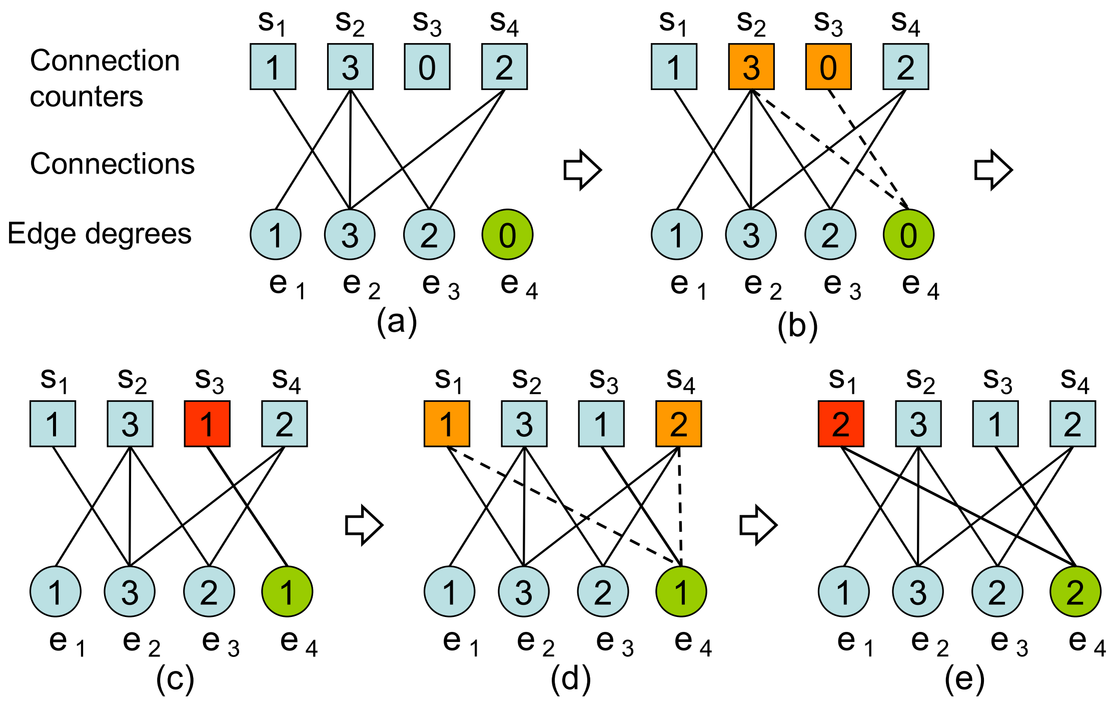


Figure 5: Procedure of tournament selection

degree of ideal soliton distribution is $\ln(k)$, and that of robust soliton is $\mathcal{O}(\ln(k/\delta))$. Furthermore, the mean degree closely connects to the computational cost of LT codes as mention in section 1. Obviously, the fundamental requirement to "fill all the bines" decisively gives the lower bound of the computational cost.

If we carefully consider the cause of missing input symbols, there is a large connection variance when random selection is adopted. For this reason, soliton distributions need to be adopted with a high mean degree to avoid missing input symbols. However, a similar effect can be obtained by reducing the variance. In order to achieve this purpose, an alternative selection strategy is proposed in this paper to greatly improve the efficiency and performance of LT codes.

3 Connection Choice

In order to stochastically equalize the connection count of each input symbol, the use of a selection strategy called *tournament selection* is introduced in this section. Tournament selection stochastically selects a source symbol with a configurable preference, parameterized as *tournament size* (T), towards those source symbols with fewer connections. Detailed operations and the effect of utilizing tournament selection are presented.

3.1 Tournament selection

In the proposed LTCC framework, random selection, which is used to select an input symbol for encoding, is replaced by tournament selection. In LTCC, the first step to select input symbols is still determining a degree d according to the adopted degree distribution. Then, tournament selection is applied to select d input symbols. At the beginning of the selection procedure for each input symbol, T, called *tournament size*, input symbols are chosen u.a.r. as connection choices. Fig. 5 illustrates a simple example step by step with tournament size $T = 2$. *Connection* *counts*, the numbers in squares, denote how many times the corresponding input symbols have been selected to encode a codeword. The connection counts of connection choices are compared, and the input symbol with the minimum connection count wins the tournament. If there is a tie, selecting one u.a.r. among the symbols with the same connection count breaks the tie. The winner is the only symbol that really takes part of encoding among the T candidates in this tournament selection run. When a new connection is chosen, the connection count of the winner is updated accordingly. The selection procedure repeats until d distinct input symbols are determined. The complete procedure is described as follows:

- Parameters
  - $(s_1, s_2, \ldots, s_k)$: input symbols
  - $(c_1, c_2, \ldots, c_k)$: connection counts
  - $\pi(d)$: degree distribution
  - T : tournament size
  - e : new encoding symbol
- Procedure to encode a codeword
  - Step 1) Decide a degree d according to $\pi(d)$
  - Step 2) For $i = 1, \ldots, d$ do
    - Step 2.1)
      Generate a random number sequence $(r_1, r_2, \ldots, r_T)$ to mark T distinct input symbols $(s_{r_1}, s_{r_2}, \ldots, s_{r_T})$ as connection choices
    - Step 2.2)
      Determine the symbol, say, s_{m_i}, that has the minimum connection count. I.e., $c_{m_i} = \min(c_{r_1}, c_{r_2}, \ldots, c_{r_T})$. If there is a tie, select one u.a.r. among the symbols with the same connection count
    - Step 2.3)
      If s_{m_i} has already been selected, discard s_{m_i} and go to Step 2.1
    - Step 2.4)
      Update the connection count $c_{m_i} = c_{m_i} + 1$
  - Step 3) Output $e = s_{m_1} \bigoplus s_{m_2} \ldots \bigoplus s_{m_d}$

Connection counts record the encoding history of input symbols. They help to identify those symbols which have fewer connections and should be selected. The use of tournament size T increases the probability for a source symbol with fewer connections to be selected, and thus, the "balls" can be relatively evenly distributed in the "bins." It can be observed that as tournament size grows, the effect to equalize the connection distribution becomes significant. Therefore, compared to LT codes, with a suitable T, LTCC can have the same missing probability for source symbols and adopt a degree distribution of which the mean degree is lower. The analysis on the missing probability of LTCC is given next.

3.2 Missing probability of LTCC

In section 2.3, the probability to miss an input symbol is evaluated for random selection. The missing probability is important and influences the design of degree distributions. Hence, such a probability for tournament selection is analyzed in this section to show the reason why LTCC possesses the features as claimed.

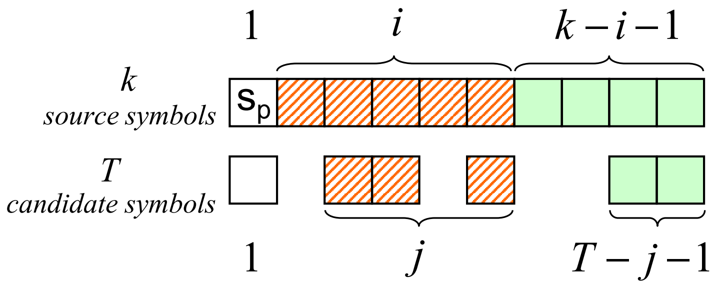


Figure 6: States during tournament selection

Let s_p denote an input symbol which has never been selected after N connections are made to the k input symbols. Here, we call s_p an *empty* symbol based on the analogy of the ball-bin problem. First of all, the probability P_i represents that a connection is made and s_p remains empty, while i input symbols are not empty. There are two situations for such an event. The simpler case is that in this selection run, s_p is not included in the T candidates, and the probability is $(k-T)/k$. In the other case, s_p is selected into the set of connection choices and remains empty. In this tie situation, the probability for s_p to remain empty depends on how many candidates are also empty. Fig. 6 illustrates a condition in which T candidates consist of s_p, j nonempty, and $T-j-1$ empty input symbols. In this condition, s_p remains empty with a probability of $(T-j-1)/(T-j)$, and as a result, the total probability in a tie situation can be calculated as a summation of all the possible combination of candidates. Equation (10) gives the complete formula to compute P:

$$\begin{aligned} P_i &= \frac{\binom{k-1}{T}}{\binom{k}{T}} + \sum_{j=a}^{b} \frac{1}{k}\binom{T}{1} \times \frac{\binom{i}{j}}{\binom{k-1}{j}}\binom{T-1}{j} \times \frac{\binom{k-i-1}{T-j-1}}{\binom{k-j-1}{T-j-1}}\binom{T-j-1}{T-j-1} \times \frac{T-j-1}{T-j} \\ &= \frac{k-T}{k} + \frac{T}{k}\sum_{j=a}^{b} \frac{\binom{T-1}{j}\binom{i}{j}}{\binom{k-1}{j}} \times \frac{\binom{k-i-1}{T-j-1}}{\binom{k-j-1}{T-j-1}} \times \frac{T-j-1}{T-j}, \\ &\text{where } \begin{cases} 0 \leqslant i \leqslant k-1 \\ a = \min(T-1, i) \\ b = \max(0, T-k+i) \end{cases}. \end{aligned} \tag{10}$$

P_i is the probability in a state that i input symbols are nonempty. Therefore, it is necessary to estimate the distribution of each state during the encoding. Let matrix Q contain entries $Q_{n,i}$ denoting the probability that there are i nonempty input symbols after n connections are made. Filling the the matrix is not difficult because there are only two states, empty or not, to consider. Obviously, the amount of nonempty symbols, i, either increases by one or does not change when a new connection is made. Suppose that q_i is the probability of no change and the event happens unless all the candidates are nonempty. q_i can be expressed with Equation (11):

$$q_i = \begin{cases} 0 & \text{for} \quad i < T \\ \frac{\binom{i}{T}}{\binom{k-1}{T}} & \text{for} \quad i \geq T \end{cases}. \tag{11}$$

Then,

$$Q_{n,i} = \begin{cases} q_i \times Q_{n-1,i} & \text{if } i = 1 \\ (1-q_{i-1}) \times Q_{n-1,i-1} + q_i \times Q_{n-1,i} & \text{otherwise.} \end{cases}. \tag{12}$$

After q_i is obtained, the entries $Q_{n,i}$ can calculated row by row with Equation (12) except for the first row. For the first connection, an empty input symbol must be chosen. Hence, $Q_{1,1}$ is initiated as 1 and the others are 0 in the first row. Then, we define

$$R_n = Q_{n,k} \times P_k \ , \tag{13}$$

and the missing probability can be calculated as

$$\prod_{n=1}^{N} R_n \ . \tag{14}$$

The product of $Q_{n,k}$ and P_k, denoted as R_n, represents the probability that s_p is not selected as neighbor for making the n-th connection. Thus, the multiplication from R_1 to R_N indicates the final probability that a missing input symbol occurs in the LTCC framework.

In order to empirically verify the derivation of the missing probability of LTCC, simulations are conducted in several different parameter settings. The calculation results and simulation data are both shown in Fig. 7 for $k = 100$. Each data point in the figure presents a mean value of 10 independent simulation runs and in each run, 100000 coding events are simulated. Hence, the attainable precision of probability is around 10^{-5}. The perfect match between the calculation results and simulation data confirms our derivation of the missing probability of LTCC. According to Equation (8), the missing probability of LT codes, δ/k, must be smaller than 10^{-4} for $k = 100$ if a probability of 0.99 that all the input symbols have neighbors is expected, and there are at least $k \ln(k/\delta)$ connections to achieve such a situation for random selection, equal to tournament selection with $T = 1$. The number of connection divided by k, $\ln(k/\delta) \cong 9.21$, can be used to roughly approximate the required mean degree for LT codes. However, in Fig. 8, we show the derived results in an extended scope. 10^{-4} can be achieved by tournament selection with $T = 2$ when the mean degree is close to 5. Furthermore, mean degrees of 3.5, 3, and 2.5 are required for tournament selection with $T = 3$, $T = 4$, and $T = 5$, respectively.

In this section, a source symbol selection strategy, tournament selection, was introduced to replace random selection in LT codes. In addition, a theoretical investigation on the missing probability of LTCC was conducted to reveal the reason why LTCC is able to deliver good performance with low computational cost. Fig. 8 shows that tournament selection is remarkably good to work with degree distributions of a low mean degree, which represents low computational cost. In the next section, the simulation results will demonstrate that not only computational cost is reduced but also less decoding overhead, the amount of extra codewords, is required to fully recover the source data.

4 Experiments

LT codes with connection choice (LTCC) proposed in this work has some different behavior in the encoding phase, compared to LT codes. In addition, soliton distributions are designed according to the required mean degree and released probability. As a result, soliton distributions are inappropriate for LTCC. New degree distributions are in need to cooperate with LTCC, because of tournament selection. However, the guideline for designing new degree distributions is still under investigation and exceeds the scope of this paper. Alternatively, we employ an optimization technique to search for good degree distributions to work with LTCC.

4.1 Search for degree distributions

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [19, 20, 21] is a well-known evolutionary algorithm for real-parameter optimization. Evolutionary algorithms solve search, design,

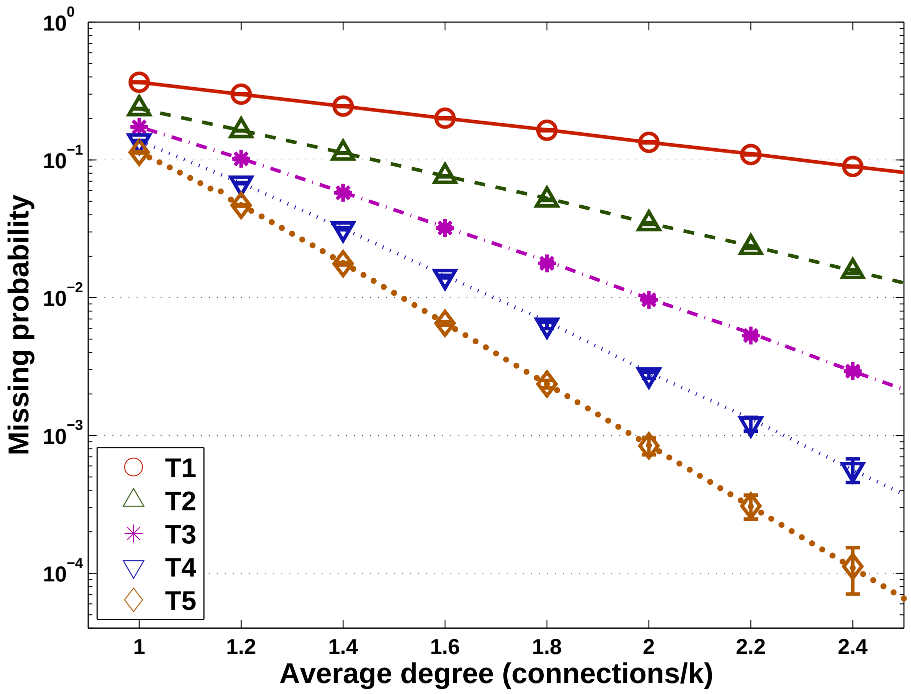


Figure 7: Probability of a missing input symbol. The lines represent the derived results calculated with Equation (14), and the markers represent the experimental results obtained from simulations.

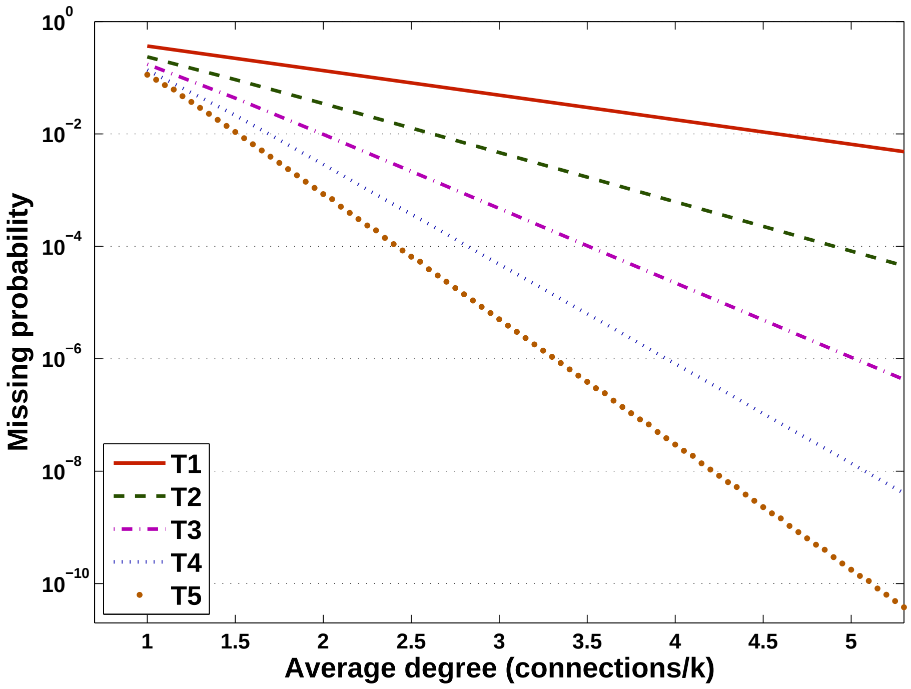


Figure 8: Probability of a missing input symbol. The lines represent the derived results calculated with Equation (14).

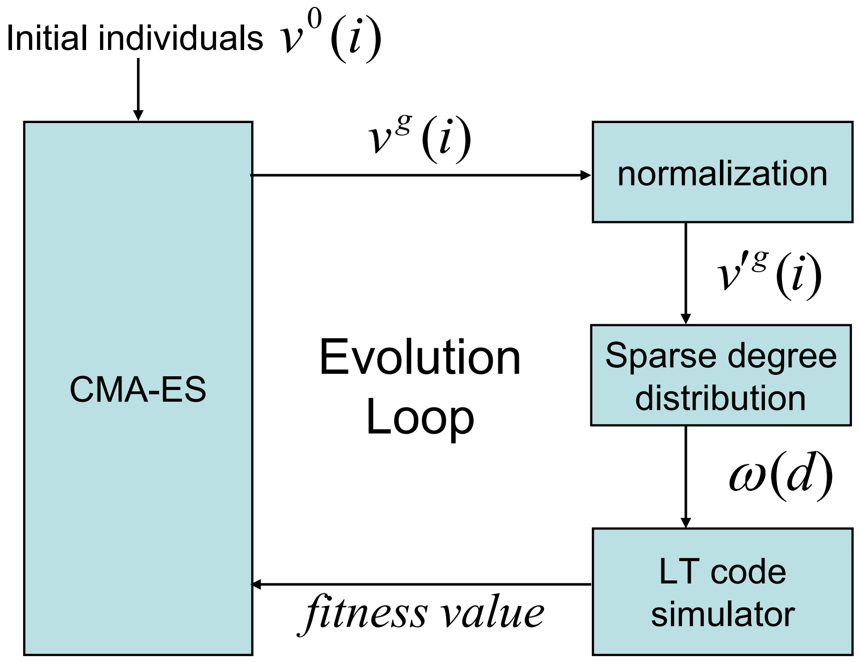


Figure 9: Procedure of employing CMA-ES to search for good degree distributions to work with LTCC.

and optimization problems by observing and mimicking natural or biological phenomena. In this work, CMA-ES is employed to search for good degree distributions not only because of its superior optimization performance but also because of its black-box property. CMA-ES works well on searching desired targets even without any domain knowledge and has been utilized to search for degree distributions in LT codes which are better than soliton distributions [22].

In order to employ CMA-ES, first of all, we need to define the decision variables. Several particular degrees called *tags* are chosen to compose the vector of decision variables. In soliton distributions, all the degrees from 1 to k are defined and have non-zero values, while there is actually no need to use all the k degrees in LTCC. With the help of tournament selection, the connections are stochastically evenly distributed, and low degrees are more important. Hence, a sparse degree distribution with only eight tags is used in this study as
Sparse Degree Distribution $\omega(d)$:

$$\omega(d) = \begin{cases} v(i) & \text{if} \quad d = tags(i) \\ 0 & \text{otherwise} \end{cases}, \tag{15}$$

where $v(i)$ is a real number vector of which the sum is 1, and tags are $\{1, 2, 3, 5, 7, 11, 13, 17\}$. The sequence composed by 1 and prime numbers is employed for an appropriate sequence density. CMA-ES takes $v(i)$, the vector of decision variables, as the individual representation and tries to search for individuals representing good sparse degree distributions. With this representation, the dimension of this optimization problem is reduced to the size of tags, and the search performance can be greatly enhanced. Notice that even though the problem search space is reduced, the correctness of our experiments is not affected. The purpose of our experiments here is to find feasible degree distributions which can cooperate with tournament selection, instead of the optimal degree distribution.

Secondly, evolutionary algorithms distinguishes individuals according to an objective function. The objective value is a quantification score which is used to measure the performance

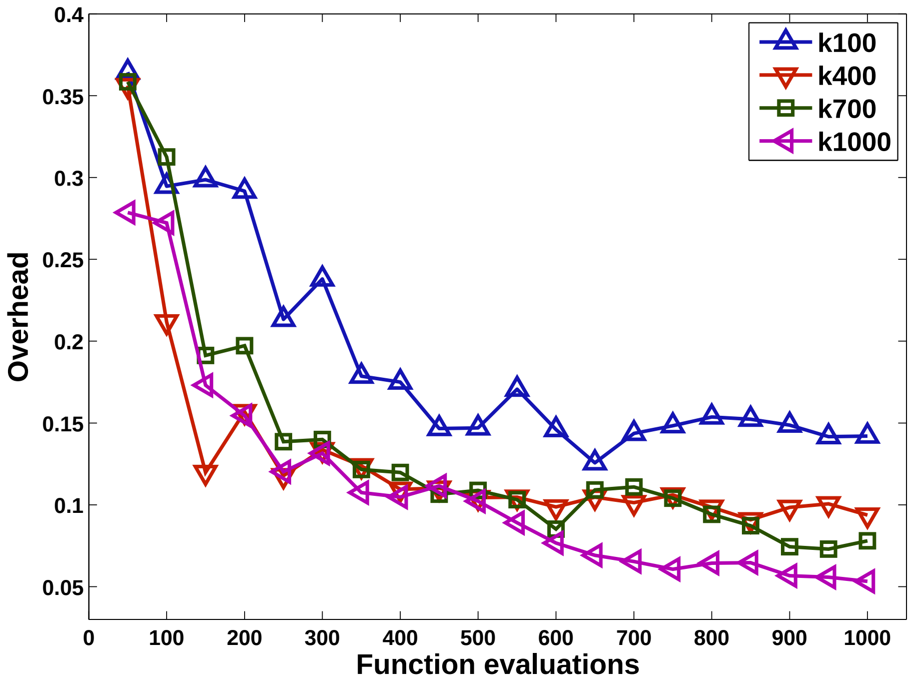


Figure 10: Optimization progress of using CMA-ES to search for LTCC degree distributions when T=3 and k=100, 400, 700, and 1000.

of individuals. The search behavior of evolutionary algorithms highly depends on how objective functions are defined. In this work, the reception overhead, ε, can be directly adopted to evaluate the performance of a given degree distribution. When a new individual is generated, LTCC simulation on the corresponding sparse degree distribution is independently repeated 100 runs, and the average reception overhead is calculated and used as the objective value for the individual.

The search procedure involving CMA-ES is shown in Fig. 9. All the individuals in the first generation are initialized with uniformly distributed random numbers, and then, each individual is evaluated by the simulator after normalization. The objective value of each individual helps CMA-ES to evolve the population in the next generation. The operation repeats until some stop condition is satisfied. In this paper, CMA-ES is employed to search for practical sparse degree distributions in the setting combinations of $T = \{1, 2, 3, 4, 5, 7, 10\}$ and $k = \{100, 400, 700, 1000\}$. The stop condition is 3000 objective function evaluations, i.e., 300000 simulation runs. Fig. 10 shows the objective changes of problems with different k's for $T = 3$. The best degree distribution during the whole process is recorded, and comparison between different settings, including the original LT codes, i.e., LTCC with T=1, is presented the next section.

4.2 Simulation results

After employing CMA-ES to search for good degree distributions to work with LTCC, we examine the LTCC performance in different setting combinations in this section. The simulation results clearly demonstrate that LTCC cannot only work with degree distributions of lower mean degrees but also require less reception overhead to fully recover the source data. Figs. 11 and 12 illustrate the required reception overhead to fully recover the source data from the aspects of different T's as well as k'. Note that LTCC with $T = 1$ is exactly the original LT codes. In Fig. 11, it is clear that reception overhead drops sharply between $T = 1$ to $T = 2$ and becomes smooth as T continues to increase except for $k = 100$. The phenomenon of the big

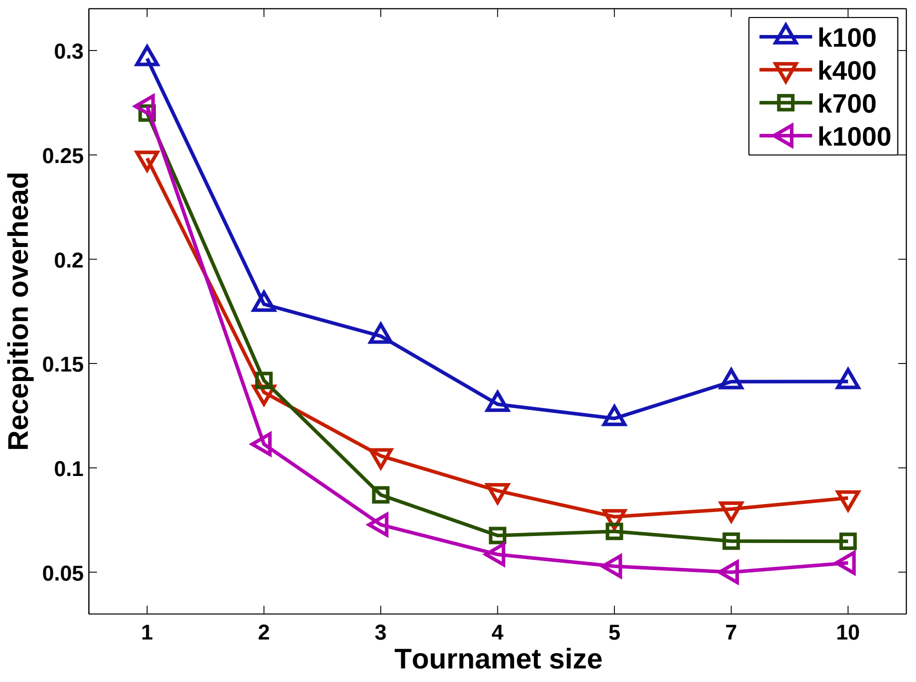


Figure 11: Reception overhead, ε, for LTCC with $T = \{1, 2, 3, 4, 5, 7, 10\}$ on $k = \{100, 400, 700, 1000\}$.

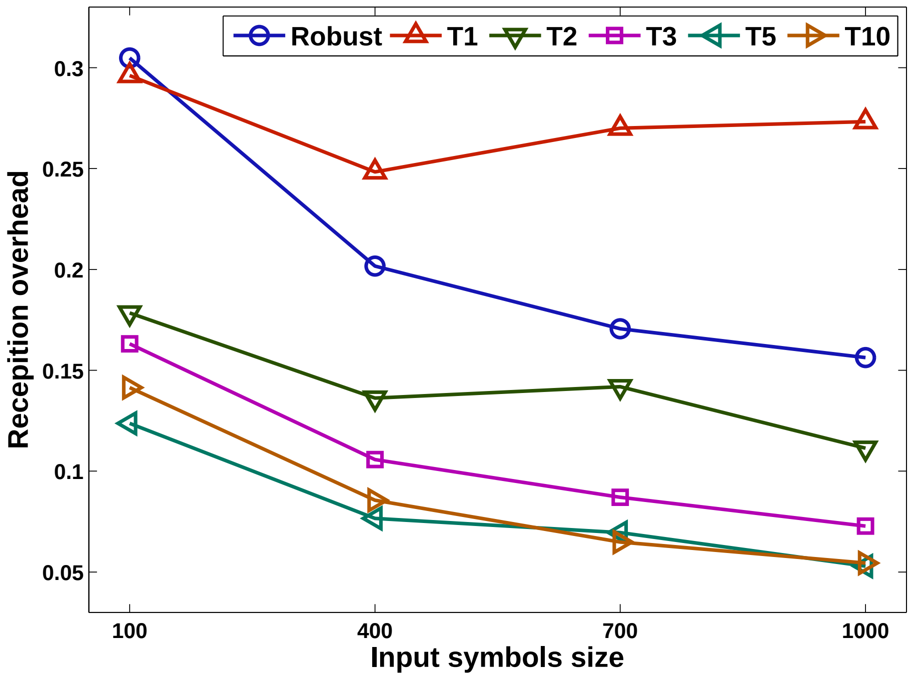


Figure 12: Comparison of reception overhead, ε, between LT codes with robust soliton distribution with $c = 0.1$, $\delta = 0.9$) and LTCC with $T = \{1, 2, 3, 5, 10\}$ on $k = \{100, 400, 700, 1000\}$.

difference between $T = 1$ and $T = 2$ indicates that LTCC greatly improves the performance, in terms of reception overhead, of LT codes by introducing tournament selection into the encoding operation. Such the low reception overhead for LT codes on small k's is very difficult, if not impossible, to obtain by adjusting parameters of robust soliton distributions. Moreover, when T increases, the improvement becomes marginal or insignificant. This indicates that in practice, T can be set to a small constant, such as 5 or 7, and be considered independent of k. As a consequence, although T is an extra parameter introduced with tournament selection into LT codes, the setting of T is not an issue. In order to show that LTCC outperforms LT codes with robust soliton distribution, Fig. 12 compares the simulation results with different T's to that obtained by LT codes with robust soliton distribution. Parameter of the adopted robust soliton distribution in this case is set as $c = 0.1$ and $\delta = 0.9$. We can observe that CMA-ES cannot find a good sparse degree distribution to work with the original LT codes, LTCC w/ T=1. However, except for T=1, LTCC outperforms LT codes with robust distributions when tournament selection operates.

The other key point of using tournament selection is to reduce the computational cost by lowering the mean degree of the adopted degree distribution. Figs. 13 and 14 show the mean degrees of the degree distributions corresponding to the simulation results presented in Figs. 11 and 12. In Fig. 13, we can observe that similar to the sharp drop between T=1 and T=2 in Fig. 11, LTCC with $T \geq 2$ can work with degree distributions with much lower mean degrees. This means that when tournament selection operates, a significant of amount of required Xor-operations can be saved. In Fig. 14, the difference between the mean degree requirement of LTCC and LT codes with robust soliton distribution increases when k increases. That is, the mean degree requirement grows more slowly for LTCC. Roughly 60% computational cost is reduced, and the ratio increases when k increases. In conclusion, LTCC requires less computational cost in order to obtain the same or better performance, in terms of reception overhead.

5 Summary and Conclusions

An improved version of LT codes, called *LT codes with Connection Choice* (LTCC), was proposed in this work. An alternative input symbol selection strategy called *tournament selection* was adopted to replace random selection originally used in the encoding phase to generate codewords in LT codes. The new parameter, tournament size, controls the variance of the number of connections to input symbols and dictates the missing probability in LTCC. Theoretical analyses on the missing probability of LT codes as well as LTCC were provided to pinpoint the reason why LTCC can work with degree distributions of lower mean degrees. Finally, experimental results obtained by conducting simulations confirmed that LTCC can achieve the same or better performance, in terms of reception overhead, with much less computational cost. In short, LT codes are an important framework of digital fountain codes, and many advanced developments depends on LT codes. There also exist data delivery protocols that employ LT codes as a functional block or fundamental component. As a consequence, according to the results obtained in this study, LTCC can be readily utilized for real-world applications, especially for k between 10^2 and 10^5, such as real-time multimedia streaming or peer-to-peer data transmission. For $k > 10^5$, although the performance of LTCC might not be significantly better than that of LT codes, the computational cost will still be greatly reduced because degree distributions with much lower mean degrees can be adopted.

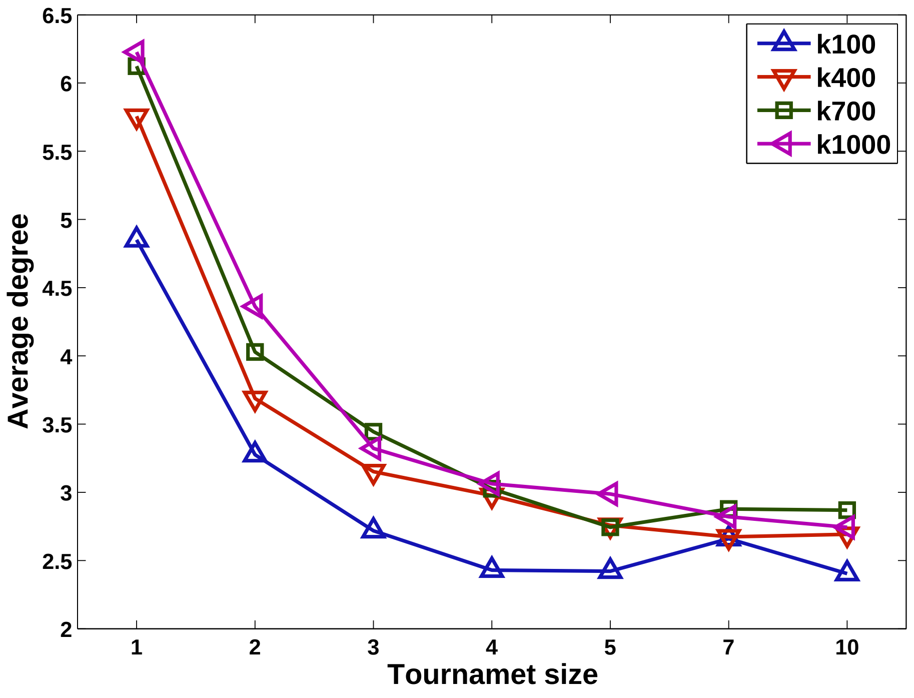


Figure 13: Mean degrees of the degree distribution optimized by using CMA-ES for the results presented in Fig. 11.

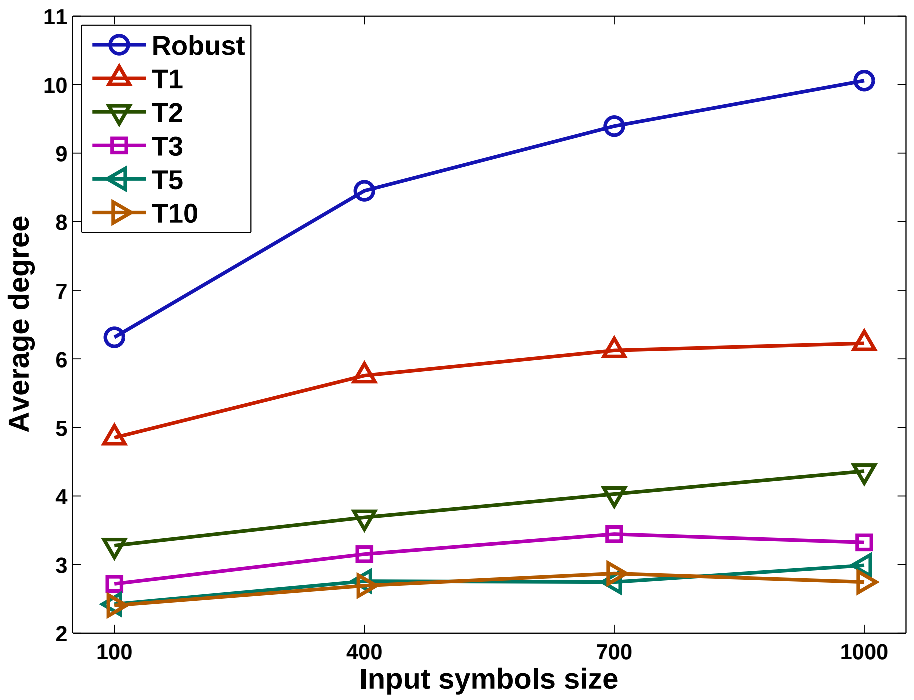


Figure 14: Comparison of mean degrees of robust soliton distribution with $c = 0.1$, $\delta = 0.9$ and the degree distribution optimized by using CMA-ES for the results presented in Fig. 12.

Acknowledgments

The work was supported in part by the National Science Council of Taiwan under Grant NSC 98-2221-E-009-072. The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

References

[1] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1998, pp. 56–67.

[2] D. J. C. MacKay, "Fountain codes," *IEE Proceedings Communications*, vol. 152, no. 6, pp. 1062–1068, 2005.

[3] M. Luby, "LT codes," in *Proceedings of the 43rd Symposium on Foundations of Computer Science*. IEEE Computer Society, 2002, p. 271.

[4] J. Pearl, "Reverend Bayes on inference engines: A distributed hierarchical approach," in *Proceedings of the American Association of Artificial Intelligence National Conference on AI*, 1982, pp. 133–136.

[5] R. Karp, M. Luby, and A. Shokrollahi, "Finite length analysis of LT codes," in *Proceedings of the IEEE International Symposium on Information Theory 2004 (ISIT 2004)*, 2004, p. 39.

[6] E. Maneva and A. Shokrollahi, "New model for rigorous analysis of LT-codes," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2006)*, 2006, pp. 2677–2679.

[7] G. Maatouk and A. Shokrollahi, "Analysis of the second moment of the LT decoder," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2009)*, 2009, pp. 2326–2330.

[8] E. Hyytia, T. Tirronen, and J. Virtamo, "Optimal degree distribution for LT codes with small message length," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM 2007)*, 2007, pp. 2576–2580.

[9] Z. Hongpeng, Z. Gengxin, and L. Guangxia, "A novel degree distribution algorithm of LT codes," in *Proceedings of the 11th IEEE International Conference on Communication Technology (ICCT 2008)*, 2008, pp. 221–224.

[10] H. Tarus, J. Bush, J. Irvine, and J. Dunlop, "Exploiting redundancies to improve performance of LT decoding," in *Proceedings of the 6th Annual Conference on Communication Networks and Services Research (CNSR 2008)*, 2008, pp. 198–202.

[11] C. Liang-Tien, L. Feng, C. Jianfei, and F. Chuan Heng, "LT codes decoding: Design and analysis," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2009)*, 2009, pp. 2492–2496.

[12] L. Michael, G. Tiago, S. Thomas, and W. Mark, "Reliable multimedia download delivery in cellular broadcast networks," *IEEE Transactions on Broadcasting*, vol. 53, no. 1, pp. 235–246, 2007.

[13] A. S. Tan, A. Aksay, C. Bilen, G. B. Akar, and E. Arikan, "Error resilient layered stereoscopic video streaming," in *Proceedings of the 3DTV Conference*, 2007, pp. 1–4.

[14] E. A. Bodine and M. K. Cheng, "Characterization of Luby transform codes with small message size for low-latency decoding," in *Proceedings of the IEEE International Conference on Communications (ICC '08)*, 2008, pp. 1195–1199.

[15] L. Haoming and I. D. Marsland, "A comparison of rateless codes at short block lengths," in *Proceedings of the IEEE International Conference on Communications (ICC '08)*, 2008, pp. 4483–4488.

[16] Z. Qian, L. Liang, C. Zeng-Qiang, and Z. Jia-Xiang, "Encoding and decoding of LT codes based on chaos," in *Proceedings of the 3rd International Conference on Innovative Computing Information and Control (ICICIC '08)*, 2008, pp. 451–451.

[17] S. Puducheri, J. Kliewer, and T. E. Fuja, "The design and performance of distributed LT codes," *IEEE Transactions on Information Theory*, vol. 53, no. 10, pp. 3740–3754, 2007.

[18] S. S. Woo and M. K. Cheng, "Prioritized LT codes," in *Proceedings of the 42nd Annual Conference on Information Sciences and Systems (CISS 2008)*, 2008, pp. 568–573.

[19] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[20] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, 2005, pp. 1769–1776.

[21] ——, "Performance evaluation of an advanced local search evolutionary algorithm," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, 2005, pp. 1777–1784.

[22] C.-M. Chen, Y.-p. Chen, T.-C. Shen, and J. Zao, "On the optimization of degree distributions in LT codes with covariance matrix adaptation evolution strategy," in *Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC 2010)*, 2010, (To appear).