A Practical Optimization Framework for the Degree Distribution in LT Codes

Chih-Ming Chen
Ying-ping Chen
Tzu-Ching Shen
John K. Zao

NCLab Report No. NCL-TR-2011001
January 2011

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/`

A Practical Optimization Framework for the Degree Distribution in LT Codes

Chih-Ming Chen, Ying-ping Chen, Tzu-Ching Shen, and John K. Zao
Department of Computer Science
National Chiao Tung University
HsinChu City 300, Taiwan
{ccming, ypchen}@nclab.tw, Stecko.cs97g@nctu.edu.tw, jkzao@cs.nctu.edu.tw

January 19, 2011

Abstract

Digital fountain is a new class of the forward error correction technology. One of the most important features is the characteristic called *rateless*, which allows encoders to generate codewords unlimitedly. As the first practical implementation of digital fountain, LT codes have been widely utilized in many areas, including multimedia streaming, long-distance communication, broadcasting systems, and distributed data storage. LT codes determine the coding structure according to a degree distribution and adopt belief propagation in decoding. Consequently, certain degree distributions are required to cooperate with belief propagation for good performance. With the proposal of LT codes, a degree distribution named *robust soliton* was presented. The capability of robust soliton distribution has been proved by theoretical analysis and asymptotically approaches optimal when the number of input symbols increases. However, most applications making use of LT codes have finite data lengths, and even some have a data length less than one thousand. Robust soliton distribution cannot guarantee satisfactory performance for relatively short data lengths. Hence, it is quite important to refine degree distributions for such cases. In this work, a practical framework which employs evolutionary algorithms is proposed to search for better degree distributions. Our experiments empirically prove that the proposed framework is robust and can customize degree distributions for different application requirements. Distributions found in the experiments are compared with a control group composed of robust soliton distributions. The performance measurements on different evaluation models are also presented to demonstrate the significant improvement of LT codes with the optimized degree distributions.

1 Introduction

In the field of communications, erasure channel is a basic transmission model in which data errors are always detected such as the connection over the internet. Once there are lost packets, the receiver must ask the sender to retransmit them. However, such a solution is inefficient if the communication works through an unreliable, low-quality channel. In order to overcome the difficulty, an alternative technique called *forward error correction* (FEC) was suggested to avoid the problem by adding redundancy information in the source data. In early days, maximum distance separable (MDS) codes were widely used to achieve FEC and could provide reliable data transmission through the erasure channel. MDS codes belong to block codes that have a fixed encoding rate. It is easy to adjust the level of fault tolerant and design specific MDS codes for different applications by deciding the code rate if we can estimate the erasure probability of the channels a priori, while MDS codes clearly do not work when various data rate are requested at the same time, for example, delivering data to multiple clients on channels of different quality.

The concept of digital fountain codes [1] was firstly proposed in 1998 and was a novel type of FEC with one very important property called *rateless*. The source data are divided into many pieces, and then different subsets of these pieces are chosen randomly to compose encoding symbols. Rateless means that the encoding process can be repeated continually. Unlimited encoding symbols are produced and sent out just like a water fountain. Any client who is interested in receiving the source data can collect these encoding symbols in an arbitrary order. The source data can be recovered as soon as a sufficient amount of encoding symbols are received. During the transmission, the encoding process of each encoding symbols is independent and unlimited, and thus, the sender does not need to handle lost packets. For this reason, digital fountain codes can conveniently solve the aforementioned problem and highly fit into the broadcasting or multicasting applications.

Luby Transform (LT) codes [2] proposed by Michael Luby in 2002 are the first practical implementation of fountain codes. There are two important features in LT codes making it practical. Firstly, the number of source data, or called input symbols, to be randomly selected to generate an encoding symbol is decided by a particular probability distribution. Secondly, a decoding approach required little computational cost is employed in the receiver side. Due to this design, the performance of LT codes totally depends on the adopted degree distribution. Base on mathematical analysis, two general guidelines to design good degree distributions are also proposed in [2]. One of them is the theoretical optimal, but impractical, design. The other is asymptotically close to optimal when the input symbol size approaches infinite. However, in practice, the source data are usually divided into a finite, or moderate, size according to different applications. Hence, efforts are needed to put in finding better degree distributions for LT codes with short input symbol sizes.

In this paper, a LT code optimization framework utilizing evolutionary algorithms is proposed to search for good degree distributions. Evolutionary algorithms are heuristic search technologies which solve problems by mimicking certain phenomena observed in nature. In our framework, degree distributions are represented as real number vectors to form the population. Each individual in the population is evaluated and compared according to a probability with which the source data cannot be fully recovered. By developing the framework, we not only expect to find degree distributions with good performance but also provide a tool that can customize degree distributions for LT-code users facing different requirements.

The remainder of this paper is organized as follows. Section 2 gives the related studies on LT codes and some work trying to improve degree distributions. Section 3 introduces the detailed coding mechanism of LT codes and the way to evaluate a degree distribution. In section 4, our optimization framework is proposed and described in detail. Several optimization experiments with different configurations are represented in section5, and the optimized distributions are compared with the original design of LT codes in section6. Finally, section 7 summarizes and concludes this paper.

2 Related Work

In this section, the development and related work of digital fountain codes are introduced. Prior to fountain codes, Reed-solomon (RS) codes [3] were the most well-known MDS codes which were used in erasure channel. Same as the other MDS codes, RS codes are a type of block codes and require quadratic decoding time. Such high complexity is unacceptable for bulk data transmission. In 1997, a precursor of digital fountain codes called Tornado codes [4] was proposed by Luby. Tornado codes encode messages based on irregular sparse graphs and reduce the decoding time complexity to linear. However, the block-code properties make them unable to satisfy the requirement of serving heterogenous clients. It was a big challenge at that time to broadcast large messages on demand. The first digital fountain protocol approximated by Tornado codes was almost immediately proposed to solve the multicast problem [1, 5], in which the requirements of an ideal multicast protocol were also explained. After that, LT codes, the first practical implementation of digital fountain, were proposed in 2002. Unlike [1], LT codes are true rateless codes which can generate arbitrary number of codewords. Not only the framework but also two degree distributions called soliton distributions were presented in the proposal. According to the theoretical analysis, k input symbols can be recovered with a successful probability $1 - \delta$ when extra $\mathcal{O}(\sqrt{k}\ln^2(k/\delta))$ encoding symbols are received. The average symbol operation, in addition, is only $\mathcal{O}(k \cdot \ln(k/\delta))$.

Even though it has not been a long time since LT codes were introduced, many related studies and applications have been proposed. Among them were Raptor codes [6, 7], the most important extension proposed by Shokrollahi in 2004. As aforementioned, the computational complexity to generate one encoding symbol in LT codes is $\ln(k)$. This is because the average degree of the adopted distribution has be sufficiently large to ensure that all input symbols are selected at least once. However, $\ln(k)$ is not practical for real-time applications such as video conferences or live TV programs. In order to solve this problem, Shokrollahi introduced a two-layer structure which added a pre-coder in front of LT codes. The added mechanism has the function to help LT codes to recover a fraction of missing symbols, and then, distributions with a constant degree average can be used for any input symbols sizes in Raptor codes. Based on similar ideas, more and more schemes involving LT codes were developed [8, 9, 10]. On the other hand, LT codes have also be applied to many real-world applications. As a branch of erasure codes, LT codes can replace traditional block codes and additionally support the rateless feature which is useful to cope with the problem of packets loss. There are hence many studies utilizing the technology in video streaming [11, 12, 13, 14]. Without heavy handshaking, LT codes are perfect to be employed in broadcasting systems [15, 16, 17]. In addition, efficiency of long distance communication can be also improved because of the reduced communication latency. Some results were presented in the areas of deep-space communication [18, 19] and satellite communication [20]. In the scenario with multiple sources, various schemes of distributed fountain codes [21, 22, 23] were proposed to reconstruct source data by receiving encoding symbols from several sources. [24] considered different levels of importance of source segments, especially for multimedia data, and a encoding mechanism with priority was introduced.

In summary, LT codes play a critical role in both new Raptor code schemes and LT code applications. Improving the performance of LT codes is therefore important if not necessary. Many studies were proposed for better performance by modifying different components of LT codes. Some [25, 26] aimed the decoding procedure and proposed different mechanisms to recover source data. They indicated a similar idea of mixing the belief propagation and gaussian elimination for balance between computational cost and overhead. [27] paid attention to the pseudo-random number generator and replaced it with a chaotic sequence. In addition to these studies, in fact, more effort has been put into the design of degree distributions in the hope that LT codes with soliton distributions might be outperformed, while robust soliton distribution is proven to be near optimal when the input symbol size approaches infinity. Thus, most topics focused on how to improve the performance of LT codes with a short data length [28, 29]. [28] even presented an approach to obtain the optimal degree distributions for $k < 30$, while applications in such a scale can easily solved by gaussian elimination [30, 31]. As a result, for the range of input symbol sizes from hundreds to ten thousands, good degree distributions are currently most in need. Aiming at this range, [32] and [33] are our preliminary studies of this paper and make the first attempt to utilize evolutionary algorithms on the optimization of degree distributions. Based on the same idea, a practical, flexible framework is introduced in this paper to enhance the performance of LT codes and to enable the use of LT codes in the range of data lengths from hundreds to ten thousands.

3 LT Codes

Before describing the proposed optimization framework, the operation of LT codes is introduced in this section as a background. Firstly, the encoding and decoding procedures are described in section 3.1. Generally, source data are divided into fragments of some identical length. These fragments are input packages or called input symbols if the length of each fragment is only a bit. For a clear presentation, the terms, input symbols and encoding symbols, are consistently used in this paper. Section 3.2 shows the two general forms named soliton degree distributions whose performance have been confirmed. Furthermore, because evaluating the quality of degree distributions is required in optimization, section 3.3 gives an efficient approach to evaluate degree distributions for LT codes.

3.1 Encoding and Decoding

Given the source data, we suppose that the source data are cut into k input symbols. Before a codeword is generated, a degree d is chosen at random according to the adopted degree distribution $\Omega(d)$, where $1 \leq d \leq k$ and $\sum_{d=1}^{k} \Omega(d) = 1$. The degree d decides how many distinct input symbols is involved to compose a codeword. d input symbols, also named *neighbors*, are then chosen uniformly at random and accumulated by XOR. In the design of LT codes, random numbers play an essential role during the encoding process. The approach employed by LT codes for a sender to inform receivers of all encoding information is achieved by synchronizing a random number generator with a specified seed.

At the receiver side, when n, which is usually slightly larger than k, encoding symbols arrive, belief propagation is used to reconstruct the source data step by step. All encoding symbols are initially covered in the beginning. For the first step, all encoding symbols with only one neighbor can be directly released to recover their unique neighbor. When a source symbol has been recovered but yet processed, it is called a *ripple* and will be pushed into a queue. At each subsequent step, ripples are popped from the queue as a processing target one by one. A ripple is removed from all encoding symbols which have it as neighbor. If an encoding symbols has only one remaining neighbor after removing, the releasing action repeats and may produce new ripples to maintain a stable size of the queue. Maintaining the size of the ripple queue is important because the decoding process fails when the ripple queue is empty and some input symbols remain uncovered. In other words, more encoding symbols are required such that the decoding process can continue. The process succeeds if all input symbols are recovered at the end of the decoding process.

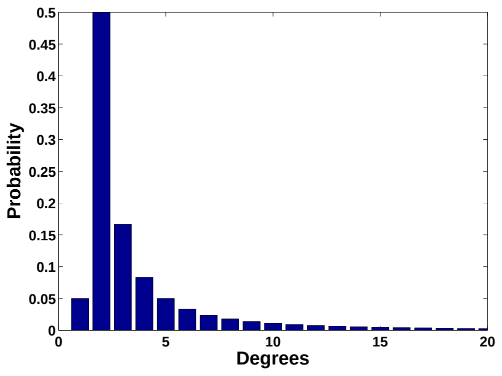
3.2 Soliton distribution

The behavior of LT codes is completely determined by the degree distribution, $\Omega(d)$, and the number of received encoding symbols, n. The overhead $\varepsilon = n/k$ denotes the performance of LT codes, and ε depends on the given degree distribution. Based on his theoretical analysis, Luby proposed the ideal soliton distribution of which the overhead is 1, the best performance, in the ideal case.

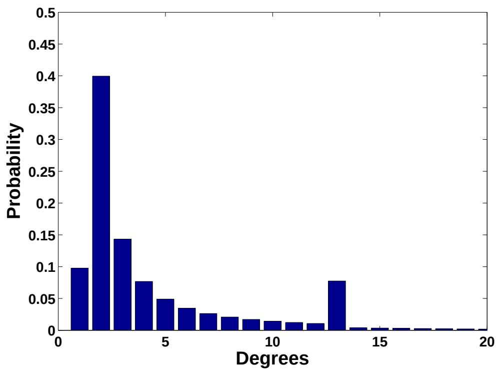
Ideal soliton distribution $\rho(d)$:

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

Ideal soliton distribution guarantees that all the release probabilities are identical to $1/k$ at each decoding step. Hence, there is exactly one expected ripple generated at each step when the encoding symbol size is k. After k processing step, the source data can be ideally recovered. Fig. 1(a) shows an example of ideal soliton distribution for $k = 20$.



(a) Ideal soliton distribution



(b) Robust soliton distribution ($c = 0.1, \delta = 0.5$)

Figure 1: Example of soliton distributions for $k = 20$.

However, ideal soliton distribution works poorly in practice. Belief propagation may be suspended by a small variance of the stochastic encoding/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. It is a baseline of ripple sizes which must be maintained to complete the decoding process. Hence, in the same paper by Luby, a modified version called *robust soliton distribution*, $\mu(d)$, was also proposed.

Robust soliton distribution $\mu(d)$:

$$S = c\ln(k/\delta)\sqrt{k}$$

$$\tau(d) = \begin{cases} S/dk & \text{for} \quad d = 1, \ldots, k/S - 1 \\ S\ln(S/\delta)/k & \text{for} \quad d = k/S \\ 0 & \text{for} \quad d = k/S + 1, \ldots, k \end{cases} . \tag{2}$$

$$\beta = \sum_{d=1}^{k} (\rho(d) + \tau(d)) \tag{3}$$

$$\mu(d) = \frac{\rho(d) + \tau(d)}{\beta} \text{ for } d = 1, \ldots, k \tag{4}$$

c and δ are two parameters for tuning robust soliton distribution. c controls the mean of the degree distribution. Smaller values of c increase the probability of low degrees, and larger values decrease the probability of low degrees. δ estimates that there are $\ln(k/\delta)\sqrt{k}$ expected ripple size as aforementioned. Fig. 1(b) is an example of robust soliton distribution with $c = 0.1$ and $\delta = 0.5$. Robust soliton distribution can ensure that only $n = k + \mathcal{O}(\ln^2(k/\delta)\sqrt{k})$ encoding 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 if k is infinite. However, in most situations, source data cannot be divided into infinite pieces, and as a consequence, the behavior of LT codes will not exactly match the mathematical analysis, especially when k is small. Furthermore, robust soliton distribution is a design for general purposes. It provides a convenient way to construct a distribution works well but not optimally. In order to obtain better performance of LT codes, the degree distribution should be customized for different input symbol sizes.

3.3 Evaluation of Degree Distributions

One intuitive indicator to evaluate degree distributions is reception overhead, ε. However, the coding process of LT codes is stochastic, and there is uncertainty in the transmission channel. A successful decoding cannot be guaranteed in the condition of a constant overhead. Instead of the average overhead, large amount of simulations are required to estimate ε. Therefore, an alternative approach used in the proposed framework is to evaluate the error probability of LT codes with some particular reception overhead. In 2004, an effective evaluation method was proposed by [34] for LT codes with a finite length. Dynamic programming was utilized to construct the distribution of ripple size during the decoding process. In short, the error probability of LT codes can be deterministically evaluated by the method when the input symbol size k and a constant overhead are given. Unfortunately, the computation considers a 3-dimensional matrix of which the edge size is k, and the complexity is very high. After applying several reduction techniques, the computational complexity is $\mathcal{O}(k^3 \log^2(k) \log\log(k))$.

After that, a new model for rigorous analysis on LT codes is proposed in 2006 [35]. The difference in this approach is to make an assumption that the number of received symbols is a random variable with mean n. For a fixed packet loss rate, it is easy to imagine that the random variable follows a binomial distribution. Based on the assumption, a fast evaluation is given as follows. First of all, for a given set of input symbols with size d, let $p_d = n\Omega_d / \binom{k}{d}$ denote the probability that an output symbol representing the sum of this set is received. Let X_u be the random variable for the expected ripple size when there are u unrecovered input symbols. Obviously, X_k is the number of input symbols which are chosen to compose the output symbols with only one degree. X_k can easily estimated as a binomial distribution $B(k, p_1)$ at beginning of decoding. At the following decoding steps $u-1$, the distribution of ripple size only depends on the previous state X_u. If $X_u = 0$, the process stops and $X_{u-1} = 0$. On the contrary, if $X_u > 0$, a symbol in the ripple queue will be removed to continue the decoding procedure and some input symbols may be released as new ripples. Let another random variable Y_u denote the number of input symbols which join the ripple at step u. Y_u also follows a binomial distribution, $B(u - X_u, q_u)$, where q_u is the probability that a symbol joins the ripple at step u. Such an input symbol a exists if there is an output symbol whose neighbors consist of symbol a, the last decoded symbol, and any set of symbols among the other $k-u$ decoded symbols. q_u can be calculated with Equation (5):

$$q_u = 1 - \prod_{d=2}^{\min(D,k-u+2)} (1-p_d)^{\binom{k-u}{d-2}} \; . \tag{5}$$

Clearly, $X_{u-1} = X_u - 1 + Y_u$. Therefore, we get Equation (6) for every r and s with $1 \le r \le u-1$ and $1 \le s \le r+1$,

$$\begin{aligned} Pr[X_{u-1} = r \mid X_u = s] &= Pr[Y_u = r - s + 1] \\ &= \binom{u-r}{r-s+1} q_u^{r-s+1} (1-q_u)^{u-r-1} \; , \end{aligned} \tag{6}$$

which gives us an expression for the distribution of X_{u-1} in terms of the distribution of X_u:

$$Pr[X_{u-1} = r] = \sum_{s=1}^{r+1} Pr[X_u = s] Pr[X_{u-1} = r \mid X_u = s] \; . \tag{7}$$

The probability that belief propagation fails to complete decoding is exactly the probability of $X_1 = 0$, which can be computed by using dynamic programming. Let $Q(u, r)$ denote $Pr[X_u = r]$,

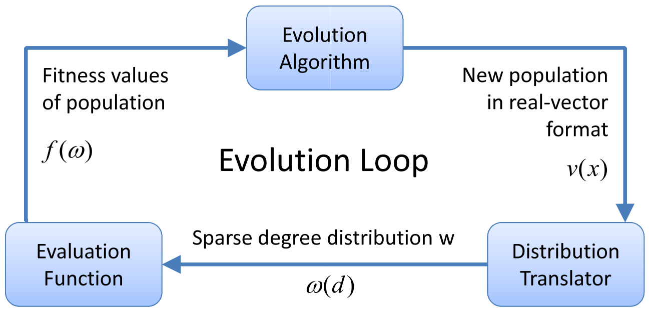


Figure 2: The flow chart illustrates basic components and evolution loop adopted in the proposed framework.

for every $u = 0, \dots, k$ and $r = 0, \dots, u$.

$$
\begin{aligned}
Q(k, r) &= \binom{k}{r} p_1^r (1 - p_1)^{k-r}, \text{ for } r = 0, \dots, k \;, \\
Q(u-1, r) &= \\
&\quad \sum_{s=1}^{r+1} Q(u, s) \binom{u-r}{r-s+1} q_u^{r-s+1} (1 - q_u)^{u-r-1}, \\
&\quad \text{for } r = 1, \dots, u-1 \;, \\
Q(u-1, 0) &= Q(u, 0) + Q(u, 1)(1 - q_u)^{u-1} \;. \qquad (8)
\end{aligned}
$$

Finally, $Q(1, 0)$ is the error probability of LT code decoding.

4 Optimization Framework

The main idea of this paper is to use optimization algorithms to search for good degree distributions for LT codes. The whole framework consists of serval components, including evolutionary algorithms, distribution translator, and evaluation function. Figure 2 shows the evolution loop of the proposed framework. The evaluation function has been previously introduced in section 3.3, and the other components will be described in this section.

4.1 Evolutionary Algorithms

Evolutionary computation [36, 37] is a branch of computational intelligence. It emphasizes to emulate various animal behavior observed in nature. For example, a population may evolve new species which can better survive when the environment changes. Ants search for food efficiently by dropping pheromone on the trail. Birds and fish get together and move in order to keep away from danger. Computer scientists were inspired from these behaviors and designed practical algorithms to solve real-world problems like searching, pattern design, and optimization. Specifically, there are many different kinds of evolutionary algorithms and they can be roughly classified according to the types of decision variables, such as bits, integers, and real numbers. The optimization target in this work is the degree distribution of LT codes which consists of probabilities represented as real numbers. Hence, three optimization approaches on the continuous domain are examined in this paper.

Evolution strategies (ES) [38, 39] are a major branch of evolutionary computation and have been developed since early 1960s. One of the simplest forms of ES is (1+1)-ES where only one child is produced by Gaussian mutation to compete with its parent in each generation, and the other is (1,1)-ES which is equivalent to random walk. Current general versions of ES are population-based and denoted as $(\mu \overset{+}{,} \lambda)$-ES. The most important property of evolution strategies is self-adaptation, a mechanism iteratively tunes the strategic parameters before the production of new offspring. The first optimization method utilized in our framework is the covariance matrix adaptation evolution strategy (CMA-ES) which was introduced by Hansen in 1996. CMA-ES is one of the most popular real-valued parameter optimization methods in evolutionary computation, and there are several variants of CMA-ES proposed in the literature [40, 41, 42]. The search ability of CMA-ES has been theoretically analyzed and empirically verified on some classic optimization problems, such as Ackley's function, Griewank's function, and Rastrigin's function. In CMA-ES, only a few algorithmic parameters need to be decided because CMA-ES can adapt parameters during the evolutionary process. Therefore, we can focus on the problem itself and employ CMA-ES as a black-box optimization tool.

Particle swarm optimization (PSO) [43] is another well-known technique for the problems composed of real decision variables. PSO was developed by Kennedy in 1995 from the inspiration of the animal social behavior, such as flocking of birds and schooling of fish. When the system is initialized, candidate solutions, called particles, are randomly distributed in the search space to represent individuals in the animal group. Then, the particles keep moving around the search space and attempt to improve the quality of solutions for some evaluation function. During the search process, each particle tracks the global best solution in the current population, the local best solution, and the personal history. While making a moving, a particle will refer to these locations to change its velocity and be guided to explore promising areas. In the past decade, PSO has been successfully applied in many domains and applications.

The other algorithm employed in this paper is called particle swarm guided evolution strategy (PSGES) [44], which a hybrid approach proposed in 2007. In the pure ES design, new offspring are generated according to a gaussian distribution and located at a mutation region with the parent as center. The mutation strength is called step size. ES can control the mutation region by adjusting the step size. Because using only one step size for all dimensions, especially when these dimensions are in different scales, is inappropriate, independent step sizes for each dimension were proposed. Moreover, a more flexible implementation named correlated mutation was also suggested to eliminate the restrictions that mutation region must be orthogonal with the coordinates. Hence, ES has the ability to freely rotate the mutation region to fit the landscape of problem. However, if there is no further information, correlated mutation just rotates blindly. PSGES introduced the mechanism of PSO to catch the global best solution to guide the direction of correlated mutation. The characteristic of PSO is used to enhance the global search capability of ES. PSGES was also verified by many well-known test functions and compared with CMA-ES on the CEC-2005 benchmark [45].

4.2 Individual Representation

An essential step of employing evolutionary algorithms is to encode the decision variables of the optimization problem. More specifically, such encoding, irrelevant to communication coding, is a function which maps solution instances of a problem onto particular representations called genotype. The relationship can be understood as various types of life are decided by their particular chromosome composition. Different encoding functions may influence the design of evolutionary operators and the size of the search space. The degree distribution is the very optimization target in the present work and can be intuitively represented as a real-number vector with length k, depending on the input symbol size. However, with such a representation, new individuals created by general-purpose evolutionary operators might not be valid probability distributions. For this reason, a normalization step is applied before the evaluation phase to

Variable	Value
A	{ CMA-ES, PSO, PSGES }
E	{ 1.05, 1.1, 1.15, 1.2 }
T	{ Full, Sparse }
K	{ 100, 400, 700, 1000 }

Table 1: The range of experimental configurations.

translate arbitrary real-number vectors into probability distributions. Such an operation is easy and does not change the feasibility of the proposed framework, although the computational complexity may be slightly increased.

Furthermore, another challenge here is the number of decision variables, i.e., the problem dimensionality. Evolutionary algorithms have been well developed to handle dimensions from ten to hundreds, even a thousand. However, a problem of higher dimensionality means a lager search space has be explored, and it is harder to discover the optimal solution. In other words, huge computational cost is required to achieve the optimization for such problems. Observing the encoding mechanism of LT codes and the belief propagation algorithm, there is no particular degree that must exist except 1, which ensures the start of the decoding process. As a result, an alternative representation called *sparse degree distribution* is suggested to reduce the number of dimensions. A sparse degree distribution has non-zero values on partial degrees, called *tags*. In fact, such a representation has been widely adopted. In this work, the tags form the vector $v(i)$ of decision variables is chosen according to the Fibonacci numbers smaller than $k/2$. The degree distribution used in this paper hence can be represented as *Sparse degree distribution* $\omega(d)$:

$$\omega(d) = \begin{cases} v(i) & d = \text{the } i\text{-th Fibonacci number, } d < k/2 \\ 0 & \text{otherwise} \end{cases}, \tag{9}$$

where k is the input symbol size of LT codes. The formula shows how to translate a real-vector into a distribution, and it is what the distribution translator does in Figure 2. There is actually no strong reason to adopt the Fibonacci numbers as tags other than that the density of the Fibonacci numbers seems appropriate. We believe that any sets of tags with similar density would also function well.

5 Experiments

In the present work, a general optimization framework that searches for good degree distributions for LT codes is proposed. In order to prove its utility and robustness, several experiments of different natures are designed and conducted to study the impact of optimization settings. These experiments can be distinguished and simply represented with a 4-tuple (A, E, T, K), where A, E, T, and K denote the different settings for algorithms, overheads, types of tags, and input symbol sizes. Table 1 shows the experimental combinations. For the first step, the optimization component is substituted with different evolutionary algorithms, including CMA-ES, PSO, and PSGES. Three algorithms are compared, and the best one will be employed for the subsequent experiments. Secondly, a constant overhead ε will be given for computing error probability in the evaluation component. Figure 3 illustrates an example curve of error probability which decreases as the reception overhead increases. Different settings of the overhead might be considered as giving the push and as expecting a minimal probability at some particular ratio of ε. Finally, a comparison between full and sparse degree distributions is given to demonstrate that such reduction is feasible and practical. Note that all the experimental results in this section are averages over 30 independent runs because of the stochastic nature of evolutionary algorithms.

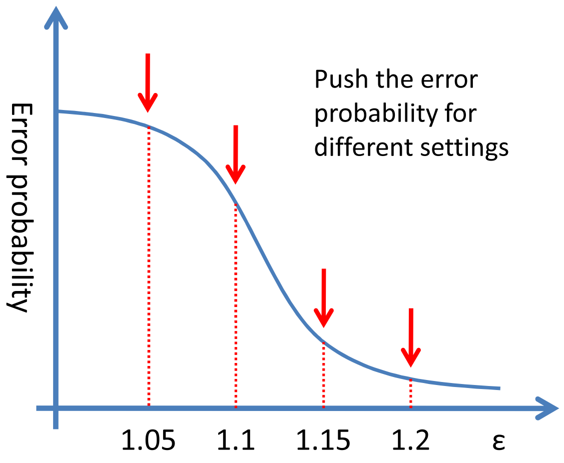


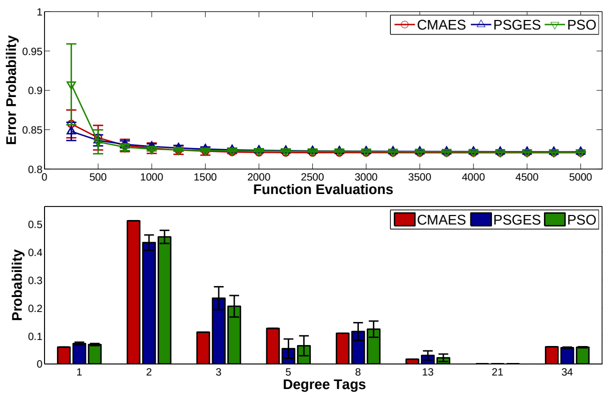
Figure 3: The figure illustrates different settings of reception overhead, ε.

5.1 Different Evolutionary Algorithms

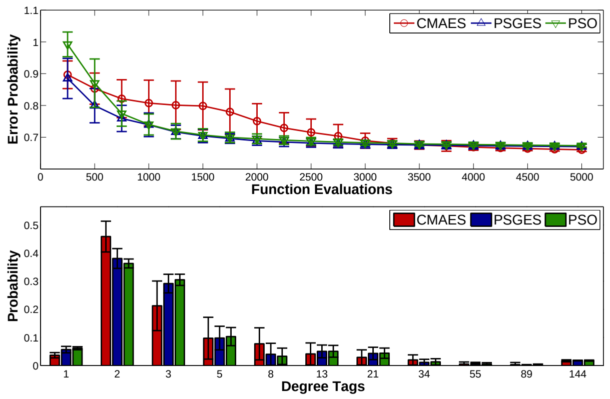
The first experiment is designed to test the performance of three evolutionary algorithms. It is in general difficult to indicate that some evolutionary algorithm is absolutely better than another because the performance of evolutionary algorithms highly depends on problem properties. As in this study, the employed objective function is a complicated computational procedure, instead of a mathematical form, the "function" is considered a black box, and we have no information regarding the landscape of the search space. We need to conduct empirical test to determine a suitable optimization component out of the three evolutionary algorithms. The experimental configuration in this part can be denoted as $A \times (1.1, Sparse) \times K$. Figure 4 shows the evolutionary process and the final results of the best degree distribution obtained. In the case of $k = 100$, three algorithms delivered very similar results, while CMA-ES found better distributions in the cases of $k = \{400, 700, 1000\}$ even though it converged more slowly. For all k sizes, the best degree distributions discovered by PSO and PSGES were similar while slight different from that found by CMA-ES. We speculate that the identical global search mechanism of PSO and PSGES makes a faster convergence and the good local search helps CMA-ES to explore deeper. According to the results, because CMA-ES can find good degree distributions with a sufficient number of function evaluations, it will be adopted in the following experiments.

5.2 Different Overhead Settings

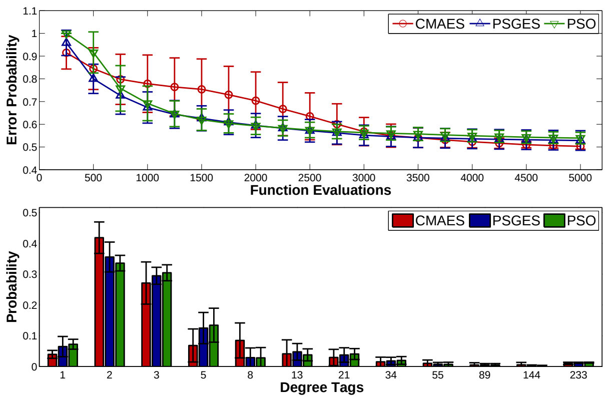
The second part of the experiments examine the different settings for the evaluation function. Figure 3 shows that the minimized target can be the error probability at any ratio of reception overhead. Totally four cases of variable E listed in Table 1 are verified. It is extremely difficult, if not impossible, to ask for a universal degree distribution which has the minimal error probability at all time. That is also why two parameters which can adjust to reflect the tradeoff between efficiency and reliability are considered in the robust soliton distribution. Figure 5 shows the results of the four experiments denoted as $(CMAES) \times E \times (Sparse) \times K$. Different degree distributions were expected for different values of ε, but the obtained degree distributions are very similar, because the average overhead for LT codes with such small input symbol sizes is much higher than $\varepsilon = 1.2$. Optimizing distributions in these situations will obtain similar results. As k increases, the obtained degree distributions are still similar, but the similarity reduces. This experiment shows that the proposed framework is robust on different optimization settings.



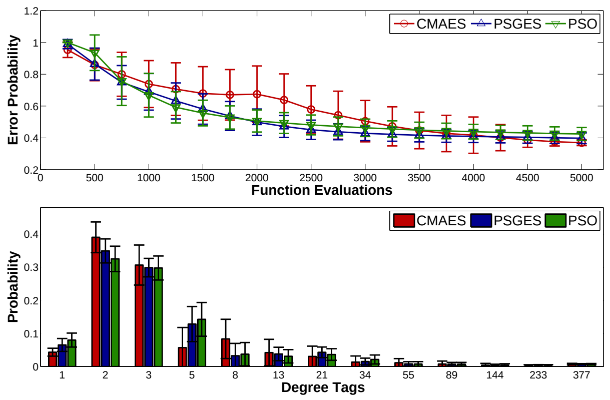
(a) $k = 100$



(b) $k = 400$



(c) $k = 700$



(d) $k = 1000$

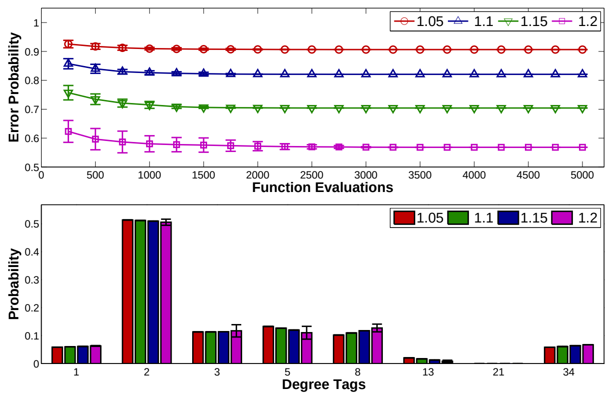
Figure 4: Optimization results obtained by using different algorithms.

5.3 Different Tags

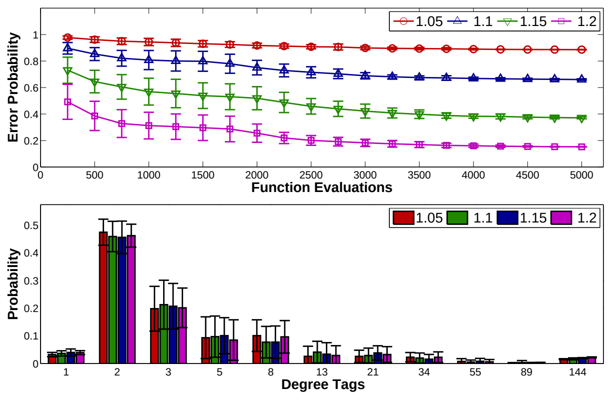
In the last experiment, the optimization of degree distributions composed of different types of tags is investigated. In order to decrease the dimensionality, the use of sparse degree distributions is suggested. Such a representation effectively reduces the search space, but the possible degree distributions are apparently also limited by not considering all the degrees. We would like to examine wether or not the effectiveness of the proposed framework is affected with this experiment. Figure 6 presents the comparison between sparse and full degree distributions. For the same reason mentioned in section 4.2, the full degree distribution is defined as a probability distribution with values at each degree from 1 to $k/2$. Only one experiment, $(CMAES, 1.1) \times T \times (100)$ was conducted because of the very high computational cost. It is clear that the optimization of full degree distributions converges slowly and reaches a slightly lower error probability in the final stage. The two obtained degree distributions seem totally different, but a series of examinations presented in the next section demonstrate their similar performance.

6 Investigation into the Optimization Results

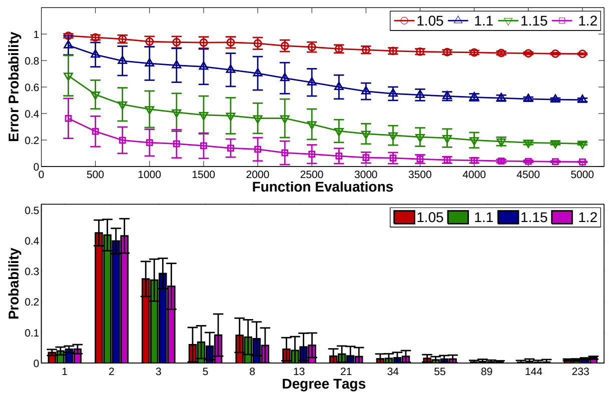
In the previous section, several degree distributions with a low error probability were found with a particular reception overhead being set. These degree distributions will be further examined on different evaluation models and compared with robust soliton distributions. Table 2 lists the best degree distributions and the corresponding error probabilities in the 30 independent runs



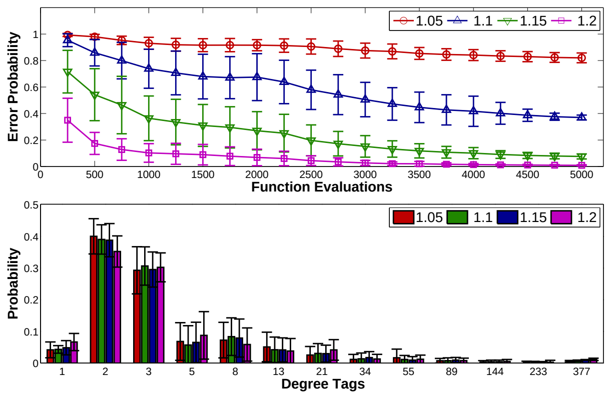
(a) $k = 100$



(b) $k = 400$



(c) $k = 700$



(d) $k = 1000$

Figure 5: Optimization results of different overhead settings.

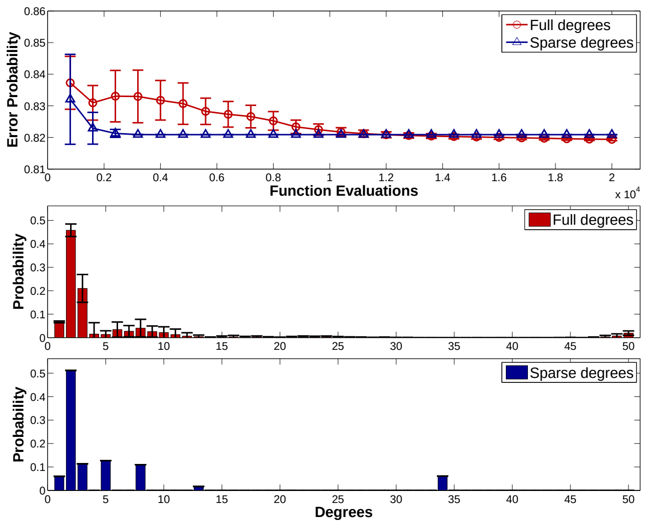


Figure 6: Optimization results of full and sparse tags are compared. More function evaluations are required for the case of full tags.

of the experiments $(CMAES, 1.1, Sparse) \times K$. In the table, the entries that are not applicable are left blank. Furthermore, some tags with a zero probability after optimization gives us an empirical proof that the use of full degree distributions is unnecessary.

Tags	k=100	k=400	k=700	k=1000
1	0.0601	0.0265	0.0249	0.0243
2	0.5120	0.5285	0.4906	0.4548
3	0.1134	0.0872	0.1560	0.2610
5	0.1269	0.2131	0.1711	0.0053
8	0.1095	0.0069	0.0202	0.1755
13	0.0169	0.0768	0.0642	0.0075
21	0.0000	0.0000	0.0312	0.0001
34	0.0611	0.0415	0.0000	0.0494
55		0.0005	0.0280	0.0000
89		0.0000	0.0000	0.0104
144		0.0190	0.0000	0.0007
233			0.0139	0.0029
377				0.0081
Error prob.	0.8209	0.6556	0.4881	0.3496

Table 2: The best degree distributions discovered in the experiments $(CMAES, 1.1, sparse) \times K$

According to the process of belief propagation, the key to successfully decoding is to maintain sufficient ripples. Therefore, the first comparison is to visualize the dynamics of ripple sizes for particular degree distributions. In the literature [2], a formula was introduced to calculate the overall release probability for a given degree distribution:

$$m(1,k) = 1$$
$$m(i,u) = \begin{cases} \frac{i(i-1)\cdot u \cdot \prod_{j=0}^{i-3} k-(u+1)-j}{\prod_{j=0}^{i-1} k-j} \\ \text{for } i = 2,\dots,k \text{ and } u = k-i+1,\dots,1 \\ 0 \text{ for all other } i \text{ and } u \end{cases}, \tag{10}$$

where $m(i,u)$ denotes the probability that an encoding symbol of degree i is released when u input symbols remain unprocessed. Hence, for a given degree distribution, $\Omega(i)$, the overall release probability $r(u)$ can be computed by the following equations.

$$r(i,u) = \Omega(i) \cdot m(i,u) \ ,$$
$$r(u) = \sum_{i=1}^{k} r(i,u) \ . \tag{11}$$

The value $r(u)$ represents the probability that an encoding symbol is released as ripple at step u. Now, it is easy to estimate the size of ripple queue if totally n encoding symbols are received. Let $R(u)$ be the average ripple size at step u and $R(k) = n \cdot r(k)$.

$$R(u-1) = R(u) + n \cdot r(u-1) - 1, \quad \text{for } u = k,\dots,2 \ . \tag{12}$$

Figure 8 presents the estimation for degree distributions given in the Table 1, in which the total received symbols are $n = k \times 1.1$, identical to the experimental configuration. The control group consists of three robust soliton distributions with different parameter settings. As Equation (3), the characteristic of robust soliton distributions is determined by the two parameters, δ and c. Luby gave the theoretical analysis [2] to show that LT codes with such

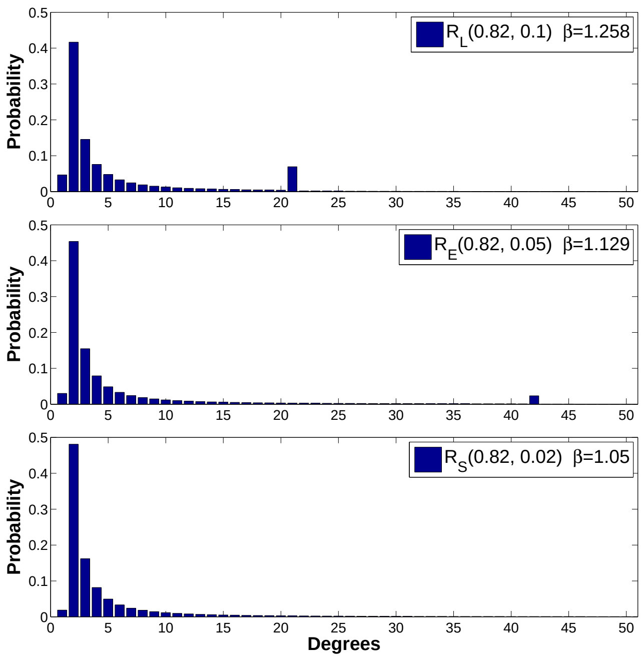
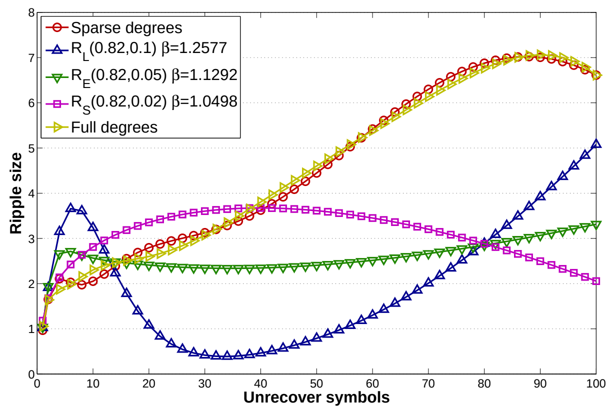


Figure 7: The control group of robust soliton distributions for $k = 100$

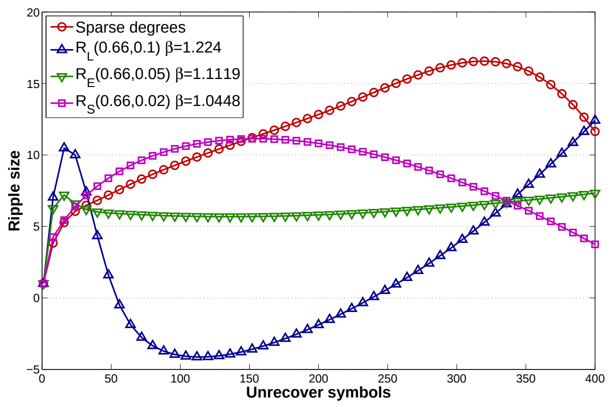
robust soliton distributions can ensure a successful decoding process with a failure probability δ when the reception overhead reaches β. Thus, the three robust soliton distributions are chosen with a fixed δ equals the optimization results and different c values to make their corresponding β satisfying the three scenarios, larger (R_L), smaller (R_S), and roughly equivalent (R_E) to 1.1. Figure 7 displays the control group of robust soliton distributions for $k = 100$.

From the dynamics shown in Figure 8, R_L (blue, up-triangle) fails to maintain enough ripples through the decoding precess when there are still around 52 uncovered symbols in the case of $k = 100$. It is no use to complete the process even though the release probability increases in the end. R_E (green, down-triangle) works well keep a stable ripple size, but the queue size is so small that it is easily be exceeded by random walk. Belief propagation may be terminated in the situation with a high probability. In the opposite of R_L, R_S (magenta,square) seems to continually accumulate ripples in the early phase of decoding and to smoothly go to the end. However, the encoding symbols with degree one of R_S are in fact too few to guarantee a successful start of belief propagation. To overcome these defects, a good decoding process has to start with a ripple queue of sufficient lengths and maintains the number of ripples as large as possible to the end. The optimized sparse degree distributions apparently satisfy the conditions and keep the number of ripples more than all the three robust soliton distributions for most of the time. In addition, the optimized full degree distribution joins the comparison for $k = 100$. A similar ripple dynamics is also obtained, and such an evidence supports that the function of full degree distributions can be approximated by sparse ones.

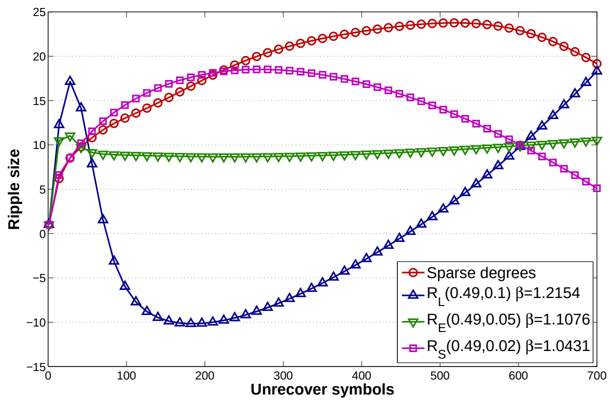
Moreover, these degree distributions are further extensively compared by using the evaluation function [35] which were introduced in section 3.3. The performance curve of each degree distribution is shown in Figure 9. With the same assumption, the curves illustrate probabilities that the receiver cannot recover source data when a particular reception overhead is received on average. The sparse degree distributions outperform robust soliton distributions in all the cases of different k's. Moreover, we can find that the performance curves of robust soliton distributions do not match the their parameter settings. It can be understood that the theoretical analysis on robust soliton distributions is based on the assumption that k is infinite. For this



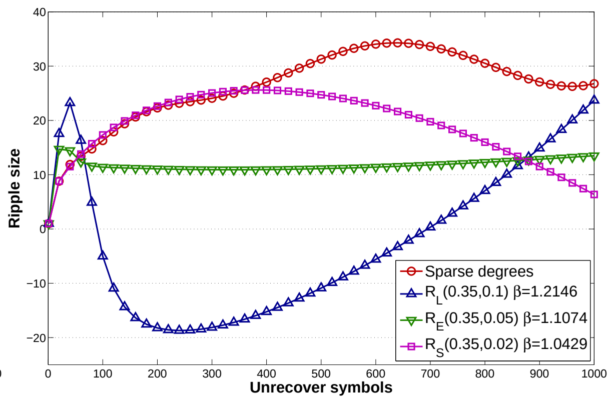
(a) $k = 100$



(b) $k = 400$



(c) $k = 700$



(d) $k = 1000$

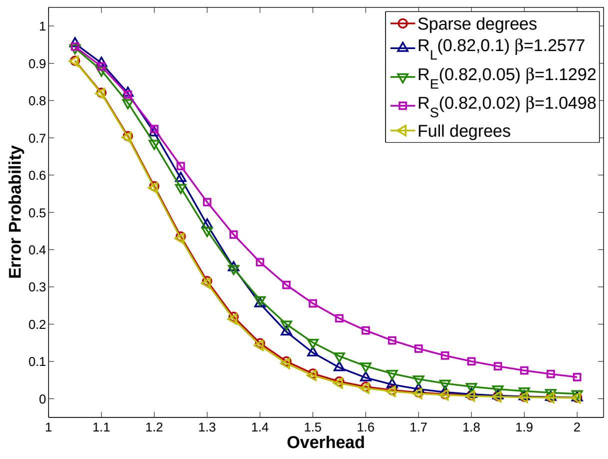
Figure 8: Dynamics of ripples. For all the k's, $\varepsilon = 1.1$

reason, there exists a gap between theory and practice for finite input symbol sizes, and the gap becomes obvious when k decreases. From the viewpoint of the receiver side, to consider the error probability of LT codes when an exact number of encoding symbols is received is more intuitive. Another evaluation method [34] based on a different assumption is also employed to investigate the degree distributions. The results are presented in Figure 10, and sparse degree distributions still have better performance than robust soliton distributions do. Even though the results of R_S is comparable at a lower reception overhead, R_S is unable to provide certain level of reliability to ensure a successful decoding process. Finally, the comparison for $k = 100$ again confirms that sparse and full degree distributions have very similar performance.

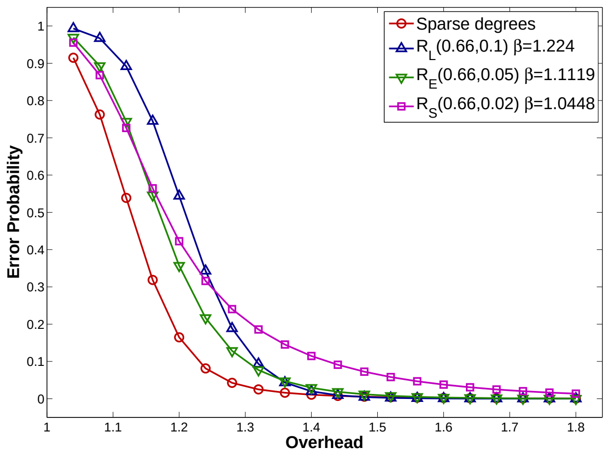
7 Conclusions

In this work, an optimization framework employing evolutionary algorithms as optimization engine was proposed to search for good degree distributions for LT codes. Firstly, several experiments were designed to verify the robustness of the proposed framework, and then, the optimized degree distributions were fully presented. These degree distributions were compared with control groups of different robust soliton distributions. It has been shown that the optimized degree distributions outperform robust soliton distributions with transmission simulations with an explanation by examining the ripple dynamics.

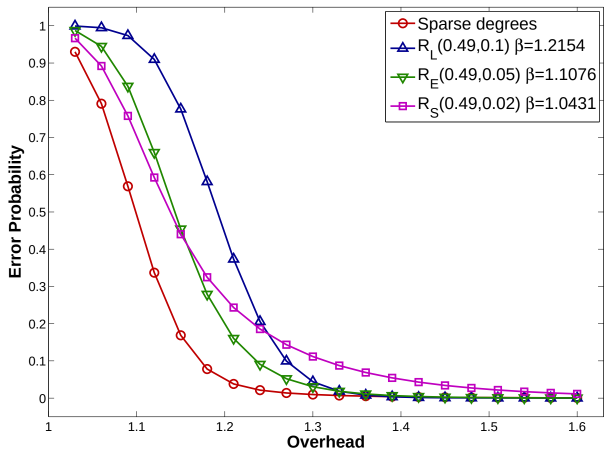
There is no doubt that LT codes are a very important implementation of digital fountain codes, and LT codes have been widely used in many areas. However, there exist a significant



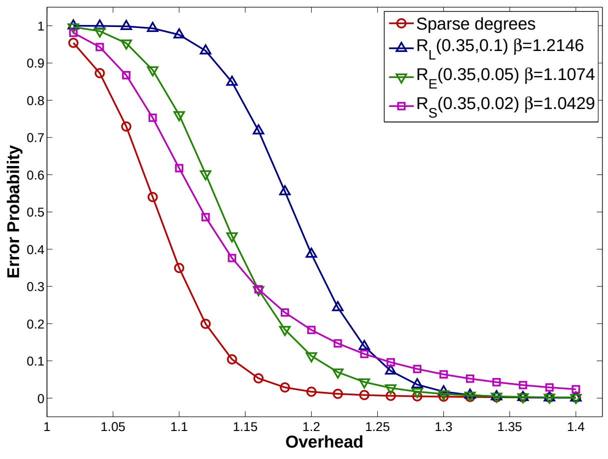
(a) $k = 100$



(b) $k = 400$



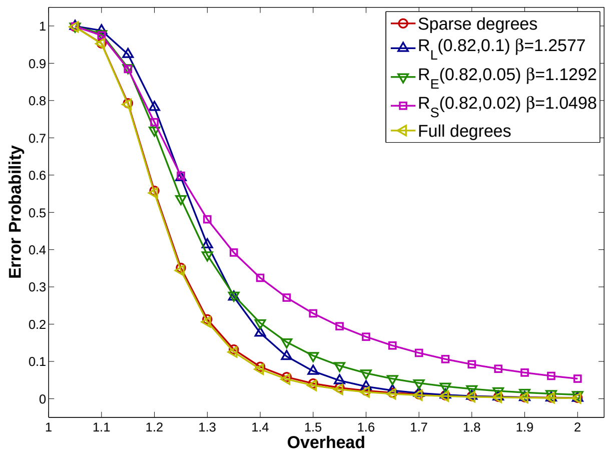
(c) $k = 700$



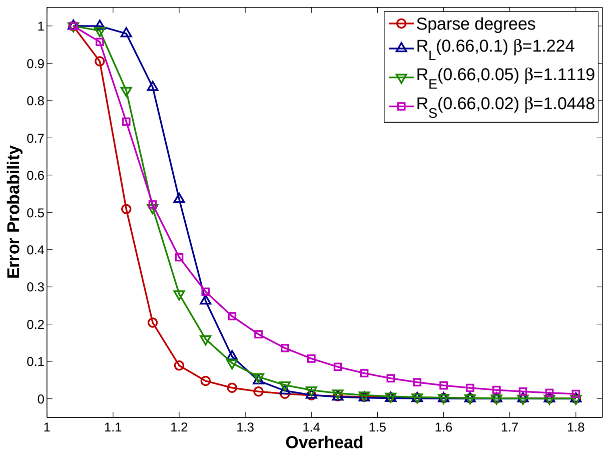
(d) $k = 1000$

Figure 9: Error probability of LT codes with the assumption that an average reception overhead is received.

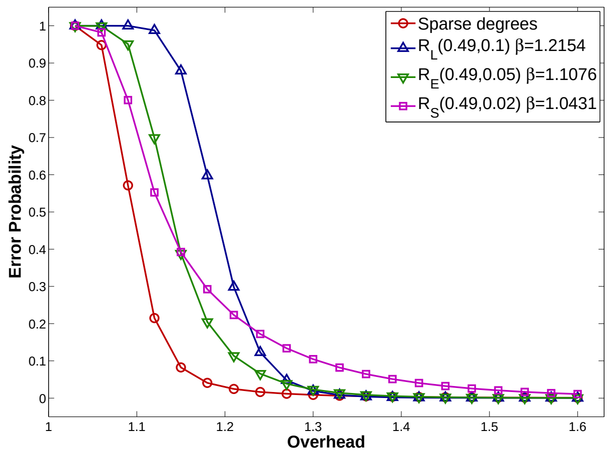
amount of applications which are limited to adopt such a technique due to the data size. Good degree distributions are necessary to boost the performance of LT codes and to support LT codes to be used in these practical applications in which the input symbol size is smaller than ten thousand. An effective and flexible solution for the issue has been proposed in the present work. The improvement makes it possible to utilize LT codes in various scenarios in the future. Moreover, the adopted evaluation function is computed with dynamic programming. It is easy to compute the error probability in the same way for 80%, 90%, or 95% input symbols are recovered. Thus, the proposed framework can also assist Raptor codes designers to customize the weaken LT code for the use of different pre-codes. Finally, some degree distributions which can provide good performance are found without mathematical analysis. Conducting theoretical investigations on these degree distributions may further our understandings of LT codes. We expect the present work to be a tool for researchers to explore more possibilities and to design better digital fountain codes.



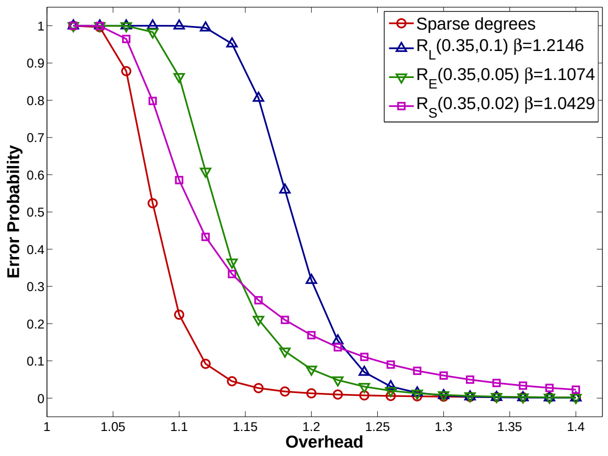
(a) $k = 100$



(b) $k = 400$



(c) $k = 700$



(d) $k = 1000$

Figure 10: Error probability of LT codes with the assumption that an exact reception overhead is received.

Acknowledgments

The work was supported in part by the National Science Council of Taiwan under Grant NSC 99-2221-E-009-123-MY2. 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.* ACM, 1998, pp. 56–67.

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

[3] I. S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[4] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 569–584, 2001.

[5] J. W. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1528–1540, 2002.

[6] A. Shokrollahi, "Raptor codes," in *Proceedings of the International Symposium on Information Theory, 2004.*, 2004, p. 36.

[7] ——, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[8] X. Yuan and P. Li, "Doped accumulate LT codes," in *Proceedings of the IEEE International Symposium on Information Theory*, 2007, pp. 2001–2005.

[9] ——, "On systematic LT codes," *IEEE Communications Letters*, vol. 12, no. 9, pp. 681–683, 2008.

[10] ——, "Quasi-systematic doped LT codes," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 6, pp. 866–875, 2009.

[11] J. P. Wagner, J. Chakareski, and P. Frossard, "Streaming of scalable video from multiple servers using rateless codes," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2006, pp. 1501–1504.

[12] M. C. O. Bogino, P. Cataldi, M. Grangetto, E. Magli, and G. Olmo, "Sliding-window digital fountain codes for streaming of multimedia contents," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 2007, pp. 3467–3470.

[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] S.-K. Chang, K.-C. Yang, and J.-S. Wang, "Unequal-protected LT code for layered video streaming," in *Proceedings of the IEEE International Conference on Communications*, 2008, pp. 500–504.

[15] M. Luby, M. Watson, T. Gasiba, T. Stockhammer, and X. Wen, "Raptor codes for reliable download delivery in wireless broadcast systems," in *Proceedings of the 3rd IEEE Consumer Communications and Networking Conference*, vol. 1, 2006, pp. 192–197.

[16] M. Luby, T. Gasiba, T. Stockhammer, and M. Watson, "Reliable multimedia download delivery in cellular broadcast networks," *IEEE Transactions on Broadcasting*, vol. 53, no. 1, pp. 235–246, 2007.

[17] Y. Ma, D. Yuan, and H. Zhang, "Fountain codes and applications to reliable wireless broadcast system," in *Proceedings of the IEEE Information Theory Workshop*, 2006, pp. 66–70.

[18] W. Yao, L. Chen, H. Li, and H. Xu, "Research on fountain codes in deep space communication," in *Proceedings of the Congress on Image and Signal Processing*, vol. 2, 2008, pp. 219–224.

[19] J. Jian, Q. Zhang, and H. Li, "Design of concatenated fountain code in deep space communication," in *Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing*, 2009, pp. 1–5.

[20] H. Zhu, G. Li, and Z. Xie, "Advanced LT codes in satellite data broadcasting system," in *Proceedings of the 11th IEEE Singapore International Conference on Communication Systems*, 2008, pp. 1431–1435.

[21] 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.

[22] C. Wu and B. Li, "rstream: Resilient and optimal peer-to-peer streaming with rateless codes," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 1, pp. 77–92, 2008.

[23] A. Oka and L. Lampe, "Data extraction from wireless sensor networks using distributed fountain codes," *IEEE Transactions on Communications*, vol. 57, no. 9, pp. 2607–2618, 2009.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] E. Hyytiä, 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.

[29] 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*, 2008, pp. 1195–1199.

[30] V. Bioglio, M. Grangetto, R. Gaeta, and M. Sereno, "On the fly gaussian elimination for LT codes," *IEEE Communications Letters*, vol. 13, no. 12, pp. 953–955, 2009.

[31] M. Rossi, N. Bui, G. Zanca, L. Stabellini, R. Crepaldi, and M. Zorzi, "Synapse++: Code dissemination in wireless sensor networks using fountain codes," *IEEE Transactions on Mobile Computing*, vol. 9, no. 12, pp. 1749–1765, 2010.

[32] C.-M. Chen, Y.-p. Chen, T.-C. Shen, and J. K. Zao, "On the optimization of degree distributions in LT code with covariance matrix adaptation evolution strategy," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2010, pp. 3531–3538.

[33] ——, "Optimizing degree distributions in LT codes by using the multiobjective evolutionary algorithm based on decomposition," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2010, pp. 3635–3642.

[34] 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.

[35] 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.

[36] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing.* SpringerVerlag, 2003.

[37] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*, 1st ed. Bristol, UK: Institute of Physics Publishing, 1997.

[38] I. Rechenberg, *Evolutionsstrategie; Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Stuttgart: Frommann-Holzboog, 1973.

[39] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies - a comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[40] A. Auger and N. Hansen, "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.

[41] ——, "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.

[42] R. Ros and N. Hansen, "A simple modification in CMA-ES achieving linear time and space complexity," in *Proceedings of the 10th international conference on Parallel Problem Solving from Nature: PPSN X.* Springer-Verlag, 2008, pp. 296–305.

[43] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[44] C.-T. Hsieh, C.-M. Chen, and Y.-p. Chen, "Particle swarm guided evolution strategy," in *Proceedings of the ACM SIGEVO Genetic and Evolutionary Computation Conference*, 2007, pp. 650–657.

[45] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization," Nanyang Technological University, Tech. Rep., 2005.