Optimizing Degree Distributions in LT Codes by Using The Multiobjective Evolutionary Algorithm Based on Decomposition

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

NCLab Report No. NCL-TR-2010002
February 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/`

Optimizing Degree Distributions in LT Codes by Using The Multiobjective Evolutionary Algorithm Based on Decomposition

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@nclab.tw, ypchen@nclab.tw, Stecko.cs97g@nctu.edu.tw, jkzao@cs.nctu.edu.tw

February 01, 2010

Abstract

LT codes are the first practical framework of digital fountain codes and have been widely used as fundamental components in many communication applications. The coding behavior of LT codes is majorly decided by a probability distribution of codeword degrees. In order to customize a degree distribution for different purposes or characteristics, a multiobjective evolution algorithm is introduced to optimize degree distributions for LT codes in this paper. Two crucial performance indicators of LT codes are considered in the present work, because minimizing the overhead of extra data packets is more important in some applications, while limiting the computational cost of the coding system in others. To flexibly handle the problem, MOEA/D is applied to optimize the two objectives simultaneously. We expect to find out the Pareto front (PF) formed by partial optimal solutions and provide the available degree distributions for different types of LT codes applications. Not only promising results are presented in this paper, but also the behavior of LT codes are thoroughly explored by optimizing the degree distribution for multiple objectives.

1 Introduction

Digital fountain codes [1] are a popular class of erasure codes in the field of communication. The concept of fountain codes was first introduced by Byers et al. [2] in 1998. Firstly, source data are divided into several pieces with an identical length. The length of each piece can be any number of bits or even several bytes. Sender generates encoded packets, or called *encoded symbols* when the packet length is one bit, by certain encoding operation. The encoding and sending procedure may repeat independently and unlimitedly. Infinite encoded packets are sent out continuously like a fountain, which is an important property of fountain codes called *rateless*. If a receiver is interested in receiving the data, it can receive the packet flow any time and collect the packets with any combination. Once sufficient packets, 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 process, no further communication is required between sender and receiver. Encoding information can be embedded in each packet. As a result, digital fountain codes are especially useful in broadcast or other situations in which back channels are unavailable. Moreover, because source data can be reconstructed no matter which packets are received, fountain codes are also considered reliable to handle the problem of packet loss.

Luby Transform (LT) codes [3] proposed by Luby in 2002 are the first practical framework and implementation of fountain codes. A novel coding mechanism based on a specifically designed degree distribution is proposed in the introduction of LT codes. The performance of

LT codes totally depends on the adopted degree distribution. In his proposal, Luby deigned a general method to construct an appropriate degree distribution to be used in LT codes, and the degree distribution was named *soliton distribution*. Via theoretical analyses, the feasibility of soliton distribution was proven [4]. Recently, researchers started to optimize the degree distribution in order to improve the performance of LT codes [5, 6], but the obtained improvement is marginal and quite limited. In these studies, only the parameters of soliton distribution were tuned and considered as decision variables, while in the present work, we directly optimize the whole degree distribution.

In the design of LT codes, redundant data and encoding computation are used to trade for the ability of forward error correction. For most applications, while the error correction ability is maintained, both costs are required to be as lower as possible, and apparently there is a trade-off among these factors. Furthermore, applications of different types and purposes have different requirements of each kind of cost. Some LT code applications which transmit data through an expensive communication channel have to reduce the data overhead. Other applications with a huge package size expect less executions of the encoding operator. In order to simultaneously satisfy those applications, multiobjectives are considered to optimize the degree distribution for LT codes in the present work. The most important motivation of this study is to fully explore the LT coding behavior with arbitrary degree distributions and to empirically provide a proof of concept that multiple requirements on LT codes can be satisfied via optimizing degree distributions with optimization techniques.

The remainder of this paper is organized as follows. Section 2 describes the detailed operations of LT codes, including the coding process and soliton distribution. Section 3 introduces the domain knowledge of multiobjective problems and the evolutionary algorithm used in this paper. Experiments and results are given in sections 4 and 5. Finally, section 6 concludes this paper.

2 LT codes

Luby introduced a new fountain code framework and gave the details of coding operation in 2002 [3]. Similar to other fountain codes, source symbols are randomly chosen to be encoded into codewords (encoded symbols). The encoding operation is achieved by a simple boolean operator, *XOR*. The relation between source data and encoded symbols can be modeled as a sparse bipartite graph. An essential design of LT codes is to decide the degree of each vertex in the bipartite graph with a probability distribution. The connectivity can be recorded as an encoding matrix and each column represents an encoded symbol. Originally, k source symbols can be fully decoding by Gaussian elimination if there exist k linearly independent columns. However, Gaussian elimination is prohibitively expensive for its computational complexity of $\mathcal{O}(k^3)$. Therefore, the belief propagation (BP) algorithm [7] is introduced to replace the expensive Gaussian elimination in the LT decoding phase. Overhead of coding is used to trade computing time because belief propagation is more efficient but more encoded symbols are needed for successful decoding. Moreover, the performance of LT codes is very sensitive to the degree distribution. A good degree distribution is necessary to co-operate with belief propagation. Luby suggested soliton distributions for LT framework in his proposal of LT codes. According to the mathematical verification, the properties of soliton distribution have been confirmed. In this section, details of coding operations and soliton distributions are described.

2.1 Encoding and decoding

Given the source data, we suppose that the source data can be cut in k source symbols with the same length of ℓ bits. Before every codeword is generated, a degree d is chosen at random according to the adopted degree distribution $\rho(d)$, where $1 \leq d \leq k$ and $\sum_{d=1}^{k} \rho(d) = 1$. The degree d decides the how many distinct source symbols will be chosen to compose an encoded symbol. d source symbols, called *neighbors*, are chosen uniformly randomly 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 the specified random number seed.

At the receiver side, when K encoded symbols were arrived which is usually slightly larger than k, belief propagation is used to reconstruct the source data step by step. All encoded symbols are initially covered in the beginning. For the first step, all encoded symbols with only one neighbor can be directly released to recover their unique neighbor. When a source symbol has been recovered but not processed, it is called a *ripple* and will be stored in a queue. At each subsequent step, ripples are popped as a processing target one by one. A ripple is removed from all encoded symbols which have it as neighbor. If an encoded symbols has only one remaining neighbor after the 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 source symbols remain uncovered. In other words, more encoded symbols are required in the decoding process. Ideally, the process succeeds if all source symbols are recovered at the end of the decoding process.

Both encoding and decoding, the LT coding operations, are achieved by *XOR*. As a result, the computational complexity of LT codes can be measured by how many times of *XOR* is executed. *XOR* operator is applied to build the connectivity in the conceptualized bipartite graph and to eliminate a ripple from the neighbors of codewords. It is evident that $d-1$ *XOR* operators are necessary to generated a codeword with degree d or uncover an encoded symbol. In the encoding phase, all encoded symbols are generated independently, and the computational complexity to produce codewords solely depends on the mean degree of the adopted degree distribution. In other words, the cost of each encoded symbol is decided by the mean of degree distributions. Hence, in practice, the mean degree is an important LT performance indicator since it presents the operational cost.

2.2 Soliton distribution

The behavior of LT codes is completely determined by the degree distribution, $\rho(d)$, and the number of encoded symbols received, K, by a receiver. The overhead $\varepsilon = K/k$ denotes the performance of LT codes, and ε depends on a 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 subsequent step. Hence, there is *one* expected ripple generated at each processing step when the encoded 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 $k = 30$.

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 that 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 the ripple queue size which must be maintained to complete a decoding process. Hence, in the same paper by Luby, a modified version called *Robust soliton distribution*, $\mu(d)$, was also proposed.
Robust soliton distribution:

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

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

c and δ are two parameters for tuning robust soliton distribution. c controls the mean of the degree distribution. Smaller value of c increases the probability of low degrees and larger one decreases it. δ estimates that there are $\ln(k/\delta)\sqrt{k}$ expected ripples as described. Fig. 1(b) is an example of robust soliton distribution with $c = 0.1$ and $\delta = 0.1$. Robust soliton distribution can ensure that only $K = k + \mathcal{O}(\ln^2(k/\delta)\sqrt{k})$ encoded symbols are required to recover the source data with a successful probability at least 1-δ.

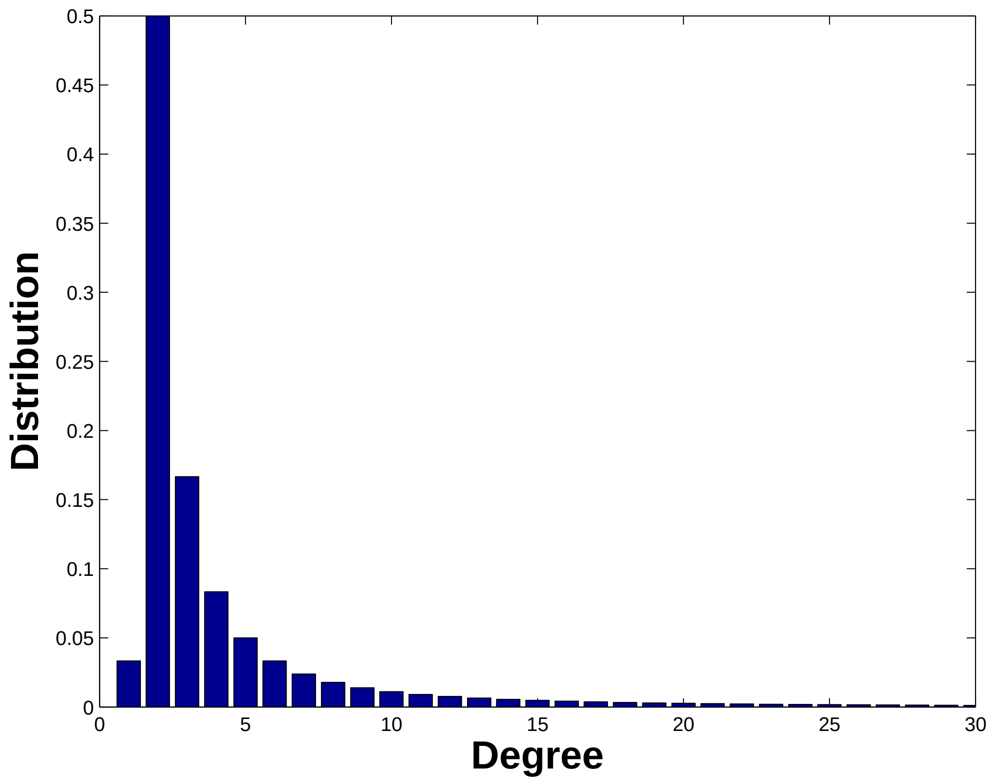
Robust soliton distribution is not only viable but also practical. The analysis of robust soliton distribution based on probability and statistics is sound if k is infinite. However, in practice, source data cannot be divided into infinite pieces, and as a consequence, the behavior of LT codes will not exactly match the mathematical analysis, especially when k is small. Furthermore, robust soliton distribution is a general purpose design. It provides a convenient way to construct a distribution works well but not optimally. In this work, we try to customize the degree distribution by using multiobjective optimization tools proposed in the field of evolutionary computation to simultaneously satisfy multiple performance requirements.

3 Multiobjective Problems

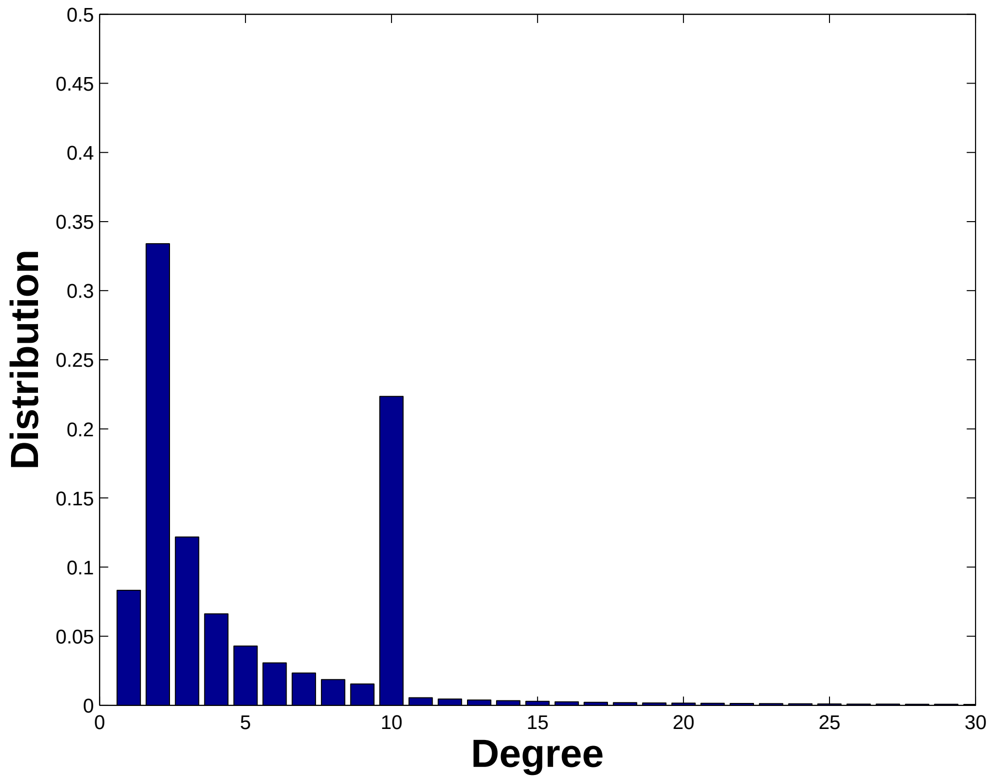
Multiobjective optimization problems (MOPs) are very important in real-world applications. There are two or more objectives should be considered simultaneously and these objectives usually conflict with each other. The most intuitive approach to deal with MOPs is to transform them into single objective problems (SOPs) by using weights on the objectives and creating a weighted sum. The approach makes the problem solvable by available tools based on mathematics or heuristics for SOPs. However, such weights oftentimes cannot be pre-determined, especially when the domain knowledge for the problem is unavailable. Furthermore, the best solution to the transformed single-objective problem is merely one solution on the Pareto front (PF) of the MOP. Hence, better optimization frameworks must be developed to fulfill the need of handling MOPs.

Due to the limitation of traditional mathematical methods for MOPs, more and more researchers try to solve MOPs in a direct way and to approximate the Pareto front as complete as possible. Their goal is to provide a set of solutions which are partially optimal. Many advanced multiobjective algorithms have been proposed in the literature. Some of them try to approximate the PF by using mathematical models, and others are developed based on evolutionary algorithms. A hybrid framework makes use of decomposition methods in mathematics and the optimization paradigm in evolutionary computation was proposed and called *multiobjective evolutionary algorithm based on decomposition* (MOEA/D) [8]. MOEA/D was proposed and shown to perform well on MOPs with complicated Pareto set shapes [9].

In this paper, we propose the use of MOEA/D to optimize the multiple objectives of LT codes. Degree distributions significantly better than *robust soliton distribution* are expected. Moreover, exploring a complete Pareto front can help researchers to analyze the trade-off between overhead



(a) Ideal soliton distribution



(b) Robust soliton distribution

Figure 1: Example of soliton distributions (k = 30)

and operational cost of LT codes. In the following section, we will give the formal description of MOPs and the MOEA/D framework, respectively.

3.1 Formal description of MOPs

In real-world applications, many problems are actually multiobjective optimization problems, and single-objective problems are a special case. A multiobjective problem can be formally stated as:

$$\begin{array}{ll} \text{minimize} & F(x) = (f_1(x), \dots, f_m(x)) \\ \text{subject to} & \begin{cases} x \in \Omega \\ C(x) = (c_1(x), \dots, c_t(x)) \geq 0 \end{cases}, \end{array} \tag{3}$$

where Ω is called the *decision space* or *variable space*, and R^m is the *objective space.* $C(x)$ represents the problem constraints and defines the feasible regions in the decision space according to problem properties [10]. $F : \Omega \to R^m$ consists of m objective functions. If Ω is a closed and connected region in R^n and all the objective functions are continuous, we call the problem a continuous MOP.

In order to consider the tradeoff between objectives, the concept of *domination* between solutions is defined. Let $u = (u_1, \dots, u_m)$, $v = (v_1, \dots, v_m) \in R^m$ be two vectors. u is said to *dominate* v if $u_i \leq v_i$ for all $i = 1, \dots, m$, and $u \neq v$. A point $x^* \in \Omega$ is *Pareto optimal* if there is no $x \in \Omega$ such that $F(x)$ dominates $F(x^*)$. The set of all the Pareto optimal points, is called the *Pareto set* (PS). The set of all the objective vectors corresponding to the PS is called the *Pareto front* (PF), where $PF = \{F(x) \in R^m | x \in PS\}$ [11].

Instead of searching for a single or just a few (Pareto) optimal solutions as in solving single-objective problems, the goal of handling multiobjective problems is to find the Pareto front as well as the Pareto set of the problem. Given the limited computational resource, including time and storage, how to provide good solutions in terms of both quality and spread is the key and challenging task for multiobjective optimization.

3.2 MOEA based on decomposition

One of the key ideas of MOEA/D is the use of a decomposition method to transform a MOP into a number of single-objective optimization problems. MOEA/D attempts to optimize these single-objective problems collectively and simultaneously instead of trying to directly approximate the Pareto front as many other evolutionary algorithms do because each optimal solution to these SOPs is a Pareto optimal solution to the given MOP. The collection of these optimal solutions is an approximation of the Pareto front. Weighted sum, Tchebycheff approach, boundary intersection, and any other decomposition approaches can serve this purpose. In the present work, the Tchebycheff approach [11] is adopted. A single-objective optimization problem obtained by decomposing the given MOP can be represented as

$$\begin{array}{ll} \text{minimize} & g(x|\lambda, z^*) = \max_{1 \leq i \leq m}\{\lambda_i |f_i(x) - z_i^*|\} \\ \text{subject to} & x \in \Omega \end{array} \tag{4}$$

where $\lambda = (\lambda_1, \dots, \lambda_m)$ is a vector of weights, i.e., $\lambda_i \geq 0$ for all $i = 1, \dots, m$ and $\sum_{i=1}^m \lambda_i = 1$. $z^* = (z_1^*, \dots, z_m^*)$ is the reference point, i.e., $z_i^* = \min\{f_i(x) | x \in \Omega\}$ for each $i = 1, \dots, m$.

Let $\lambda^1, \dots, \lambda^N$ be a set of N weight vectors. If we use a large N and select the weight vectors properly, all the optimal solutions of the SOPs transformed from decomposition will well approximate the Pareto front. Moreover, we can define a neighborhood relationship for each SOP by computing Euclidean distances between weight vectors. SOPs which are considered neighbors are assumed to have similar fitness landscapes and their optimal solutions should be

Table 1: Parameter settings of MOEA/D

Parameter	Value
N	50
T	10
Max Gen.	150

close in the decision space. MOEA/D exploits the information sharing among SOPs which are neighbors to accomplish the optimization task effectively and efficiently. The specification of MOEA/D is stated as follows:

- Inputs:
  - decision variables.
  - objective functions.
  - N: the number of subproblems.
  - T: the number of neighbors for each subproblem.
  - stopping criteria.
- Outputs:
  - Approximation to the *PS*: $x^1, \ldots, x^N$.
  - Approximation to the *PF*: $F(x^1), \ldots, F(x^N)$.

4 Experiments

The experiment implementation is described in this section. MOEA/D is a well-developed tool and has the characteristic of black-box optimization like other evolutionary algorithms. As described in section 3.2, only input and output should be handled properly. Section 4.1 shows how to encode a degree distribution into decision variables, and the objective functions are given in section 4.2. Table 1 lists the other algorithmic parameter settings of MOEA/D.

4.1 Decision Variables

The first step to use an evolutionary algorithm is to encode the decision variables of the optimization problem. It is not difficult in this study because a degree distribution can directly form a real-valued vector. In the evaluation phase, a real-valued vector of arbitrary values can be interpreted as a probability distribution, i.e., a degree distribution, with normalization. Such an operation does not change the feasibility, although the problem complexity may be slightly increased. The definition of degree distributions tells us that $d \leq k$. For a specific source symbol size k, obviously the problem dimensions is at most k. However, according to the LT encoding/decoding operations, we usually do not need non-zero probabilities on every single degree. Observing the soliton distribution and considering the belief propagation algorithm, there is no necessary degree except 1, which ensures the start of belief propagation. As a result, we optimize a selected subset of degrees in the present work. We choose some particular degrees, $\{$*1,2,3,4,5,7,9,13,17,23*$\}$ to form the decision variables according to the experience. Different subsets of degrees may change the numerical results of experiments results, but the soundness of this paper will be not be affected.

4.2 Objectives

In this paper, degree distributions are optimized for two different objectives. The first indicator to evaluate efficiency of LT codes is overhead ε. The redundancy are traded for the benefit of fountain codes and those extra encoded symbols increase the cost when they are transmitted to the receiver. In most application, overhead is required to be as low as possible because the transmission is usually expensive. In our simulation of LT codes, encoded symbols are provided unlimitedly until source data are fully recovered. The average required codewords are calculated as the fitness. The other objective is the computational cost of the encoding and decoding process. Such an objective value can be estimated with the mean of degree distributions. If M_d denotes the mean value of a degree distribution, the number of how many times XOR is executed can denote as $(M_d - 1) * \varepsilon$. There is a trade-off between ε and M_d because when M_d is greater, fewer encoded symbols may be required, and therefore, ε is less. On the other hand, M_d is the operational cost, which is the average number of XOR operators that have to be executed.

5 Experimental Results

For an n-objective problem, a solution can be represented as a point in the n-dimensional space, and all the non-dominated solutions form a partial optimal set called the Pareto front. The mission of multiobjective algorithms is to approximate the Pareto front. Hence, the solutions should well spread to provide sufficient choices to the decision marker. In our experiments, overhead and operational cost of LT codes are minimized together. Both the minimal value of objectives are expected. Clearly, a degree distribution with the minimal operational cost has only non-zero probability on degree one because in such a case, no encoding operation is needed. The case is the pure transmission without any channel coding, and it is a special case in the LT code framework. As regards the other objective, overhead has a lower bound at ratio 1. Each encoded symbol can generate a new ripple to recover a source symbol ideally such that at least k encoded symbols are required to reconstruct the original data. Different from the operational cost, such a degree distribution is not yet discovered and even its existence is not proved. Figs. 2, 3, and 4 show the optimization process and final results for k=100, 300, and 500, respectively. After 150 generations, a significant PF is represented by fifty individual points. The solution with the minimal operational cost in expectation has been found, but the best overhead is 1.2068. Example individuals are listed in Table 2 and simulated in Fig. 5, where the best value of overhead and operational cost are presented in columns 2 and 3. Columns 4 and 5 give the average overhead and execution counts of XOR in numerical simulations. Fig. 5 demonstrates the degree distribution and simulation results.

To our limited knowledge, there is no guideline to design a robust soliton distribution for some particular coding behavior. In order to fairly compare our optimized results with that of robust soliton distribution, MOEA/D is also applied to optimize the parameters of robust soliton distribution, which are c and δ. The PF of optimized robust soliton distributions is presented in Fig. 7. In the dimension of operational cost, optimized robust soliton distributions deliver very similar results because robust soliton distributions can also become the degree distribution with only non-zero probability on degree one if the appropriate parameters are given. However, there are significant differences in the other dimension. The performance are quite limited, and such a situation is caused by the fixed formula of robust soliton distribution. The figure demonstrates numerous better degree distributions that are very different from the design of robust soliton distribution. There degree distributions can be discovered by optimization algorithms proposed in the realm of evolutionary computation.

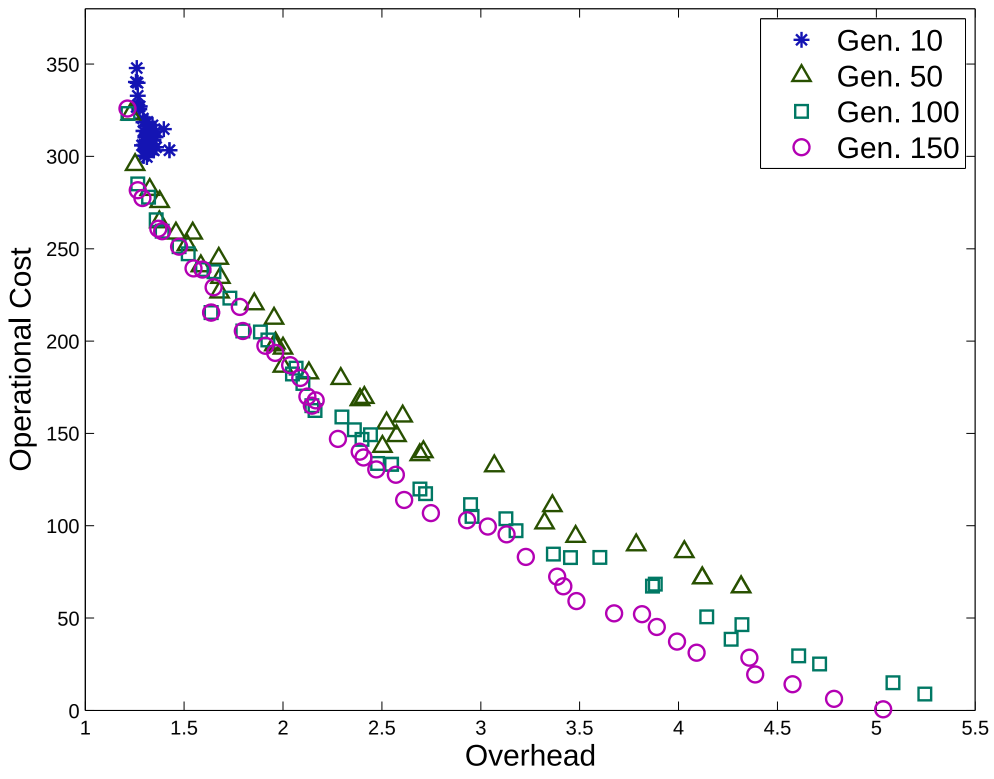


Figure 2: Evolutionary process during the optimization for $k = 100$

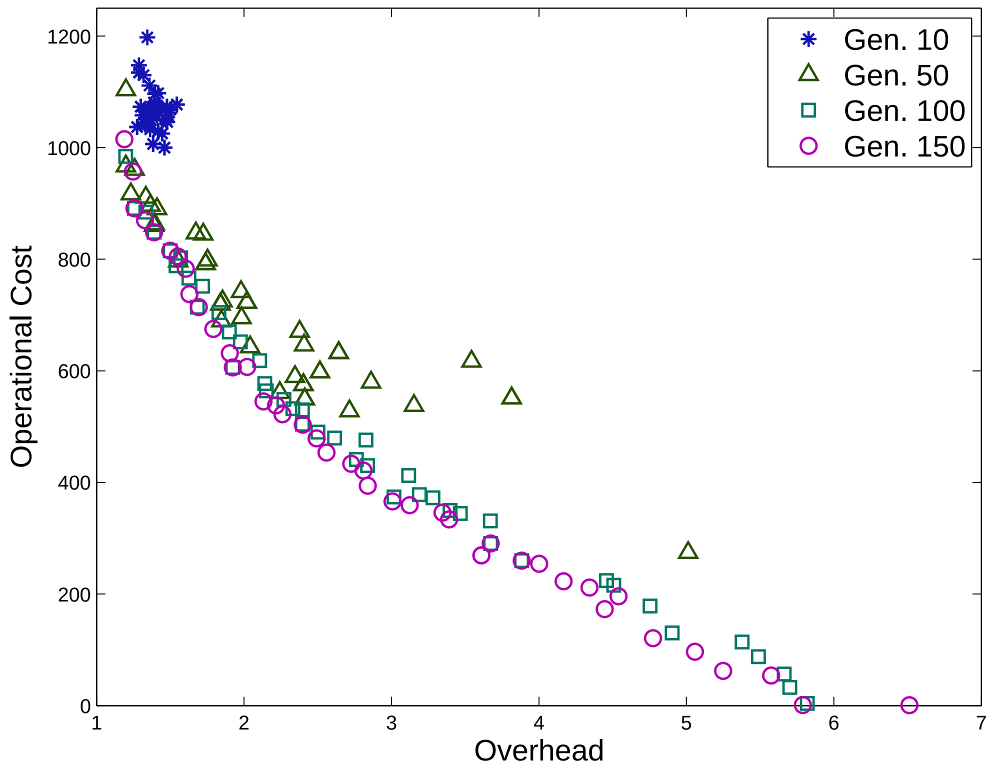


Figure 3: Evolutionary process during the optimization of $k = 300$

Table 2: Optimized arbitrary degree distributions ($k = 100$)

Individual	Best Overhead	Best Cost	AVG. Overhead	XOR
1	4.8442	0.00042	5.1958	0.038
25	2.5608	1.29873	2.6655	407.026
35	2.0294	1.85193	2.1485	558.667
45	1.4564	2.5135	1.57211	603.742
50	1.2068	2.93541	1.2718	843.669

Table 3: Optimized robust soliton distributions ($k = 100$)

Individual	Best Overhead	Best Cost	AVG. Overhead	XOR
1	4.8080	0.00552	5.1323	0.377
25	3.1244	1.74314	4.1662	125.455
35	2.0708	2.76115	2.6217	324.580
45	1.5194	3.41297	1.9278	471.753
50	1.2530	6.71008	1.3097	1141.46

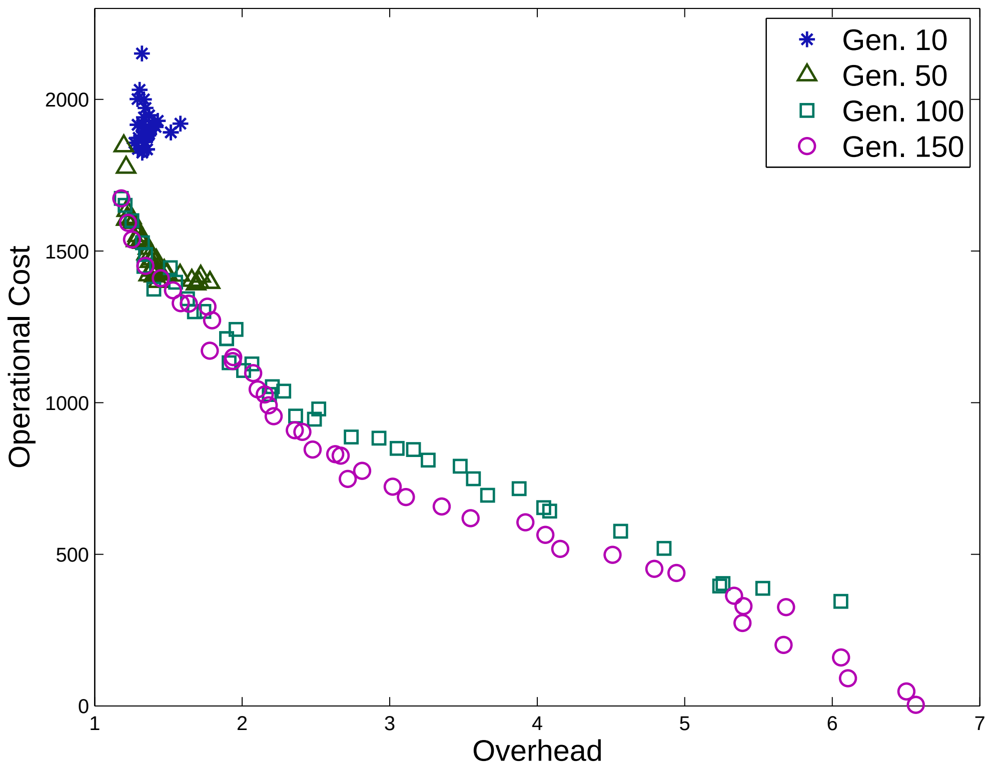
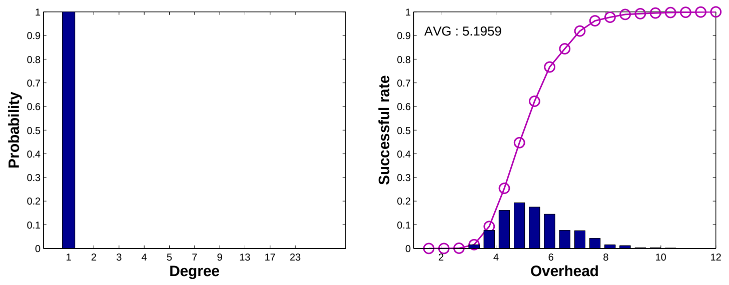
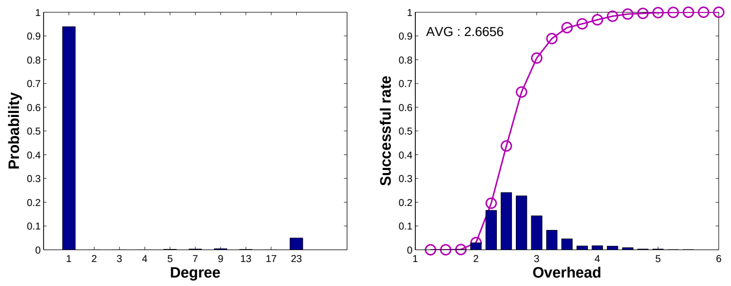


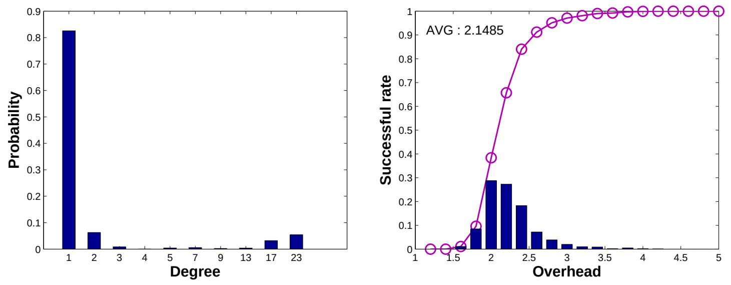
Figure 4: Evolutionary process during the optimization of $k = 500$



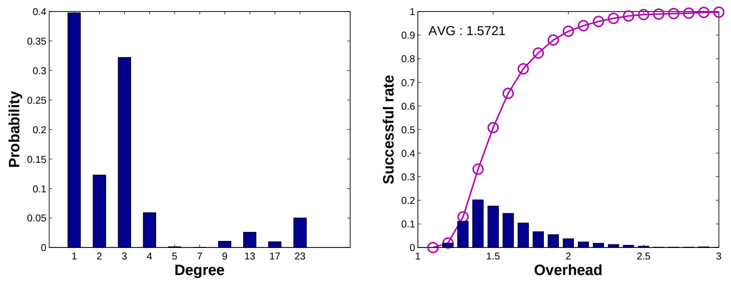
(a) Individual 1



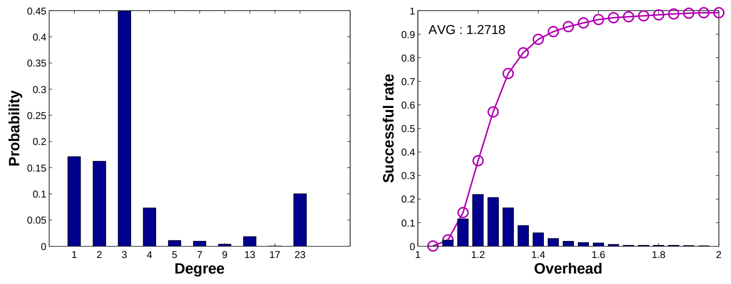
(b) Individual 25



(c) Individual 35

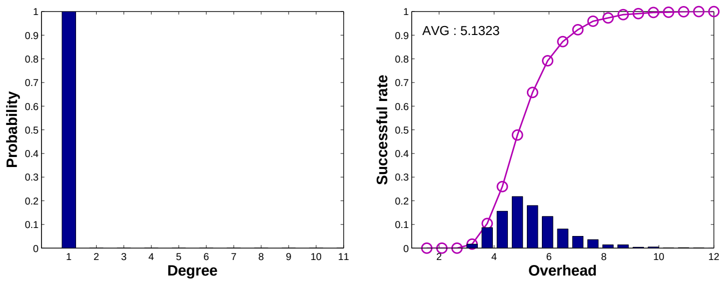


(d) Individual 45

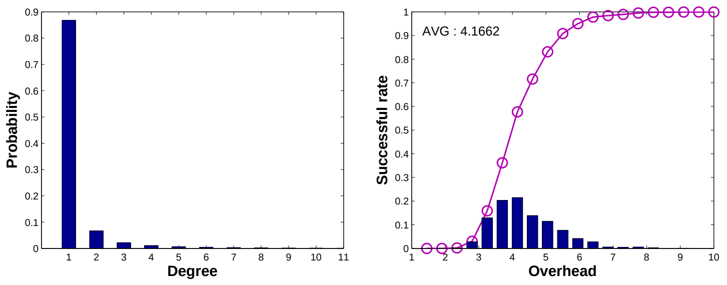


(e) Individual 50

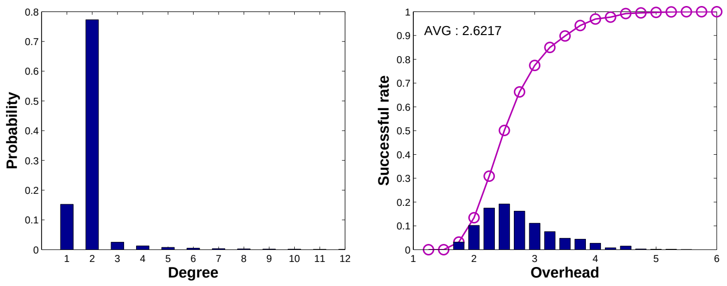
Figure 5: Simulation results of optimized arbitrary degree distributions



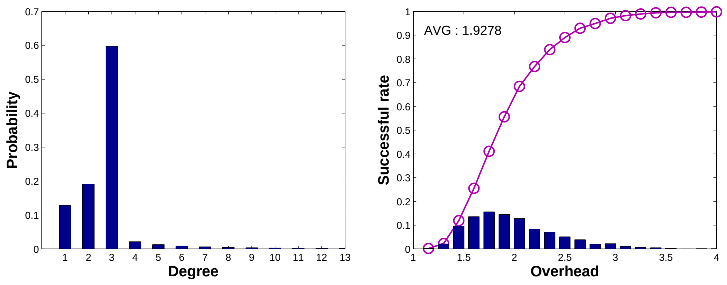
(a) Individual 1, $c = 9.72$ and $\delta = 0.00107$



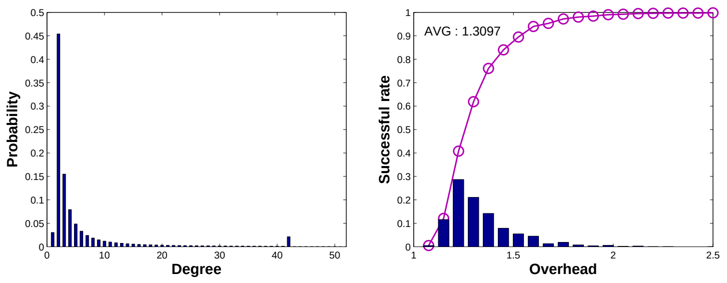
(b) Individual 25, $c = 1.634$ and $\delta = 0.185$



(c) Individual 35, $c = 2.146$ and $\delta = 0.978$



(d) Individual 45, $c = 0.96$ and $\delta = 0.601$



(e) Individual 50, $c = 0.0521$ and $\delta = 0.931$

Figure 6: Simulation results of optimized robust soliton distributions

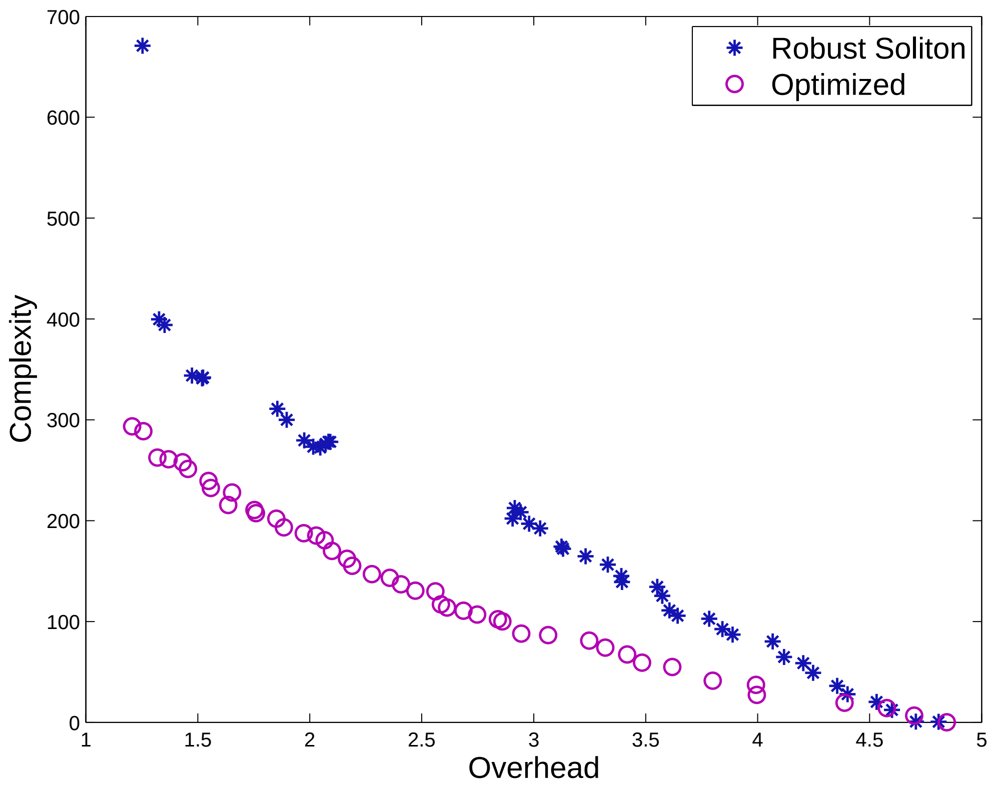


Figure 7: Comparison between the optimized arbitrary degree distribution and robust soliton distribution

6 Conclusions

This paper proposed the use of multiobjective evolutionary algorithms to optimize the degree distribution in LT codes. Overhead and operational cost were considered as two objectives and optimized simultaneously by using MOEA/D. The experimental results were promising and indicated that the Pareto front was well described. These results might also help researchers to better understand the behavior of LT codes. For applications of different types and natures, LT codes will be more efficient if choosing a specifically appropriate degree distribution is possible. Not only more choices of degree distributions are available, but also much better performance than that delivered by robust soliton distribution can be achieved, because most robust soliton distributions are dominated by the solutions discovered with MOEA/D in the experiments.

An alternative approach which may be better than robust soliton to design a degree distribution for LT codes is given in this study. While LT codes are already employed in real-world communication apparitions, such as 3GPP (mobile phone specification), the degree distribution is proven able to be customized to satisfy different requirements by evolutionary algorithms. The suitability of degree distributions will much enhance the performance of those applications. Moreover to better understand the behavior of LT codes will help the improvement of LT codes. Our final results show that some better degree distributions are beyond the model of robust soliton distribution. The theoretical analysis will be conducted on these newly discovered distributions in our future work. An advanced model for degree distributions of which the performance approximates the Pareto front can be expected.

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] D. J. C. MacKay, "Fountain codes," in *The IEE Seminar on Sparse-Graph Codes.* IEE, 2004, pp. 1–8.

[2] 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.* Vancouver, British Columbia, Canada: ACM, 1998, pp. 56–67.

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

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

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

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

[7] 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.

[8] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[9] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto set, moea/d and nsga-ii," *IEEE Transactions on Evolutionary Computation*, 2008, in press.

[10] K. Deb, A. Pratap, and T. Meyarivan, "Constrained test problems for multi-objective evolutionary optimization," in *First International Conference on Evolutionary Multi-Criterion Optimization.* Springer Verlag, 2001, pp. 284–298.

[11] K. Miettinen, *Nonlinear Multiobjective Optimization.* Kluwer Academic, 1999.