On the Optimization of Degree Distributions in LT Codes with Covariance Matrix Adaptation Evolution Strategy

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

NCLab Report No. NCL-TR-2010001
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/`

On the Optimization of Degree Distributions in LT Codes with Covariance Matrix Adaptation Evolution Strategy

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 have been a popular and practical technique in the field of channel coding since their proposal. The key component of LT codes is a degree distribution which is used to determine the relationship between source data and codewords. Luby in his proposal suggested a general method to construct feasible degree distributions. Such a general design works appropriately in typical situations but not optimally in most cases. To explore the full potential of LT codes, in this work, we make the first attempt to introduce evolutionary algorithms to optimize the degree distribution in LT codes. Degree distributions are encoded as real-valued vectors and evaluated by numerical simulation of LT codes. For applications of different natures, two objectives are implemented to search good degree distributions with different decoding behavior. Compared with the original design, the experimental results are quite promising and demonstrate that the degree distribution can be customized for different purposes. In addition to manually adjusting the degree distribution as the common practice, the work presented in this paper provides an efficient alternative approach to use and adapt LT codes for both practitioners and researchers.

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 bits or even several bytes. Sender generates encoded packets, or called encoded symbols when the packet length is one bit, by certain particular 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 channel 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 is 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 in the literature [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 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 consider the degree distribution itself as our decision variables.

Base on LT codes, an improved framework call *Raptor codes* [7, 8] was proposed by Shokrollahi. Shokrollahi integrated LT codes with a pre-coding layer. Compared with pure LT codes, the design of Raptor codes requires a degree distribution, called *weakened LT*, with some very different behavior and properties. Several instances were given in [9] for certain particular sizes of source symbols, but there is no existing guidelines regarding how to construct suitable degree distributions for the other sizes. For this regard, we demonstrate the use of optimization techniques proposed in evolutionary computation for obtaining degree distributions of different, desired properties.

In this paper, according to our limited knowledge, we make the first attempt to utilize evolutionary computation techniques to optimize the degree distribution for LT codes and demonstrate the feasibility of customizing degree distributions for different purposes. Particularly, we adopt the covariance matrix adaptation evolution strategy (CMA-ES) [10] to directly optimize degree distribution for two goals: reducing the overhead and lowering the failure rate. The experimental results are remarkably promising and show that significantly reduced overheads and lower failure rates can be achieved for LT codes with the obtained degree distribution for a wide range of source symbol sizes.

The remainder of this paper is organized as follows. Section 2 describes the detailed operations of LT codes include the coding process and soliton distribution proposed by Luby. Section 3 introduces the evolutionary algorithm used in this paper. Experiments and results are given in section 4. Finally, section 5 concludes this paper.

2 LT codes

Luby introduced a new fountain code framework and gave the detail 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. A critical change in LT codes is to decide the degree of each vertex in the bipartite graph with a probability distribution. The connectivity can be recorded as a 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 [11] 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 deciding 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.

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 an 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 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:

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

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

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 ripple size 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 the successful probability at least 1-δ.

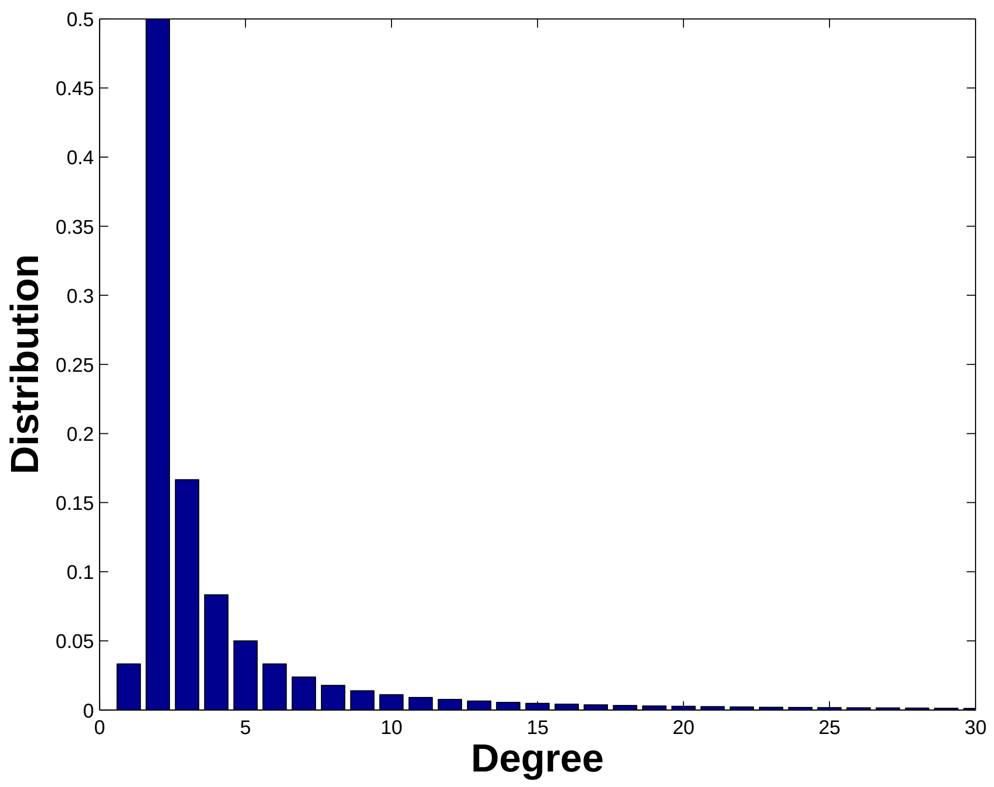
Robust soliton distribution is not only viable but also practical. The analysis of robust soliton distribution based on probability and statistics is sound if k is infinite. However, in practice, source data cannot be divided into infinite pieces, and as a consequence, the behavior of LT codes will not 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 optimization tools proposed in the field of evolutionary computation.

3 Optimization Method

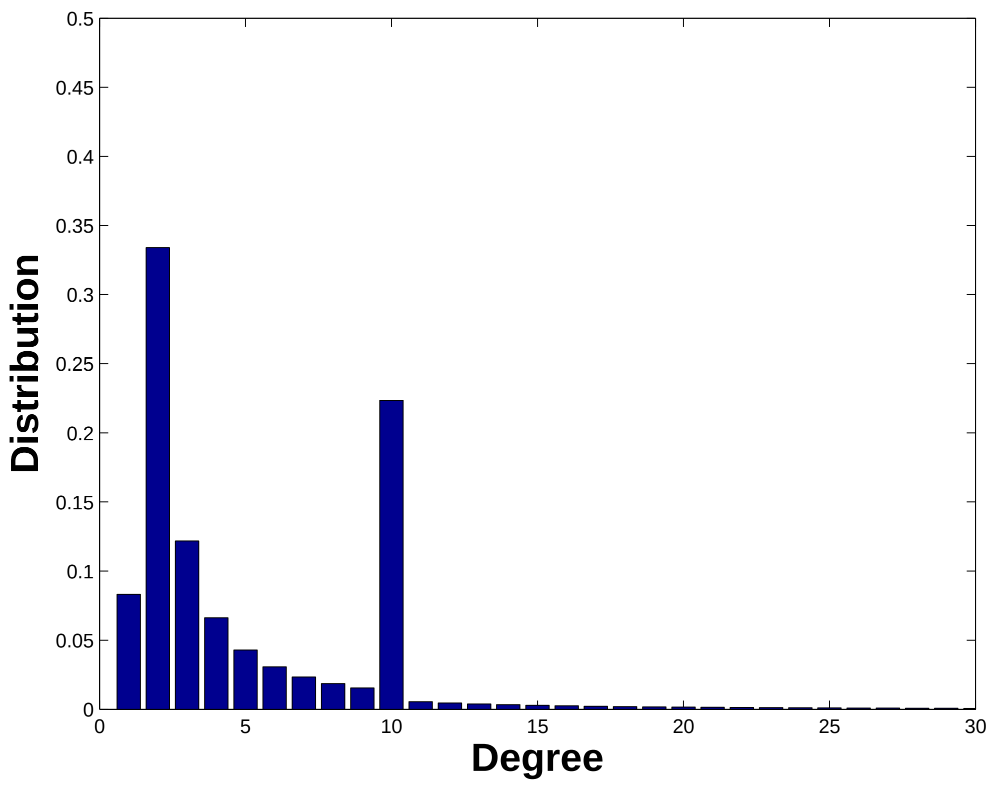
Evolution strategies (ES) are a major branch of evolutionary computation and have been developed since early 1960s. The key idea of ES is to evolve strategic parameters as well as decision variables. ES is well-known quite capable of dealing with continuous optimization problems. One of the simplest 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 denoted as $(\mu \overset{+}{,} \lambda)$-ES. The covariance matrix adaptation evolution strategy (CMA-ES) [10] was firstly introduced by Hansen in 1996 and is one of the most popular real-parameter optimization methods in evolutionary computation. There are some variants of CMA-ES proposed in the literature [12, 13, 14]. The search ability of CMA-ES has been theoretically analyzed and empirically verified on certain 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 inherits the mechanism to adapt strategic parameters during the evolutionary process. In this work, CMA-ES is utilized to optimize the degree distribution in LT framework for a wide range of k, the size of source symbols. In the remainder of this section, the way to adopt CMA-ES to handle the optimization of degree distributions are presented in detail.

3.1 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-number vector. In the evaluation phase, a real-number 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 distributions 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 degrees called *tags* to form the vector $v(i)$ of decision variables according to the Fibonacci numbers smaller than



(a) Ideal soliton distribution



(b) Robust soliton distribution

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

half of k. A degree distribution used in this paper hence can be represented as the following formula.
Optimized degree distribution $\omega(d)$:

$$\omega(d) = \begin{cases} v(i) & d = \text{the } i\text{-th Fibonacci number, } d < k/2 \\ 0 & \text{otherwise} \end{cases} . \qquad (3)$$

3.2 Objectives

We try to use two indicators to evaluate degree distributions for LT codes in this paper. The first one is the efficiency of the LT code with the optimized degree distribution which has been discussed in section 2.2. ε denotes the expected rate of overhead to transmit data. For example, $\varepsilon = 1.2$ means that in addition to the size of source data, 20% extra data are needed to recover the complete source data. This objective is to obtain some degree distribution for a specific k with the smallest ε. LT codes are rateless, and the coding process depends on randomness and probability. Source data recovered by a fixed amount of encoded symbols cannot be guaranteed. Therefore, in order to evaluate ε, we provide infinite encoded symbols, in the form of a stream of encoded symbols, to simulate the decoding process until all source data are recovered. The average of required encoded symbols per simulation is the fitness value of degree distributions.

The second indicator is the number of source symbols that cannot be recovered when a constant ratio of encoded symbols are received. In raptor codes, Low-density-parity-check (LDPC) [15] is introduced as a second layer pre-coding into LT codes. LDPC is a kind of forward error correction codes, and more information can be found in [16, 17]. It can fix errors of data without extra information as long as the errors rate is lower than certain restriction. In such a condition, the mission of LT codes is no longer to achieve full decoding. Instead, most of source symbols can be recovered with a small overhead is sufficient. For this purpose, we try to minimize the number of un-recovered source symbols given a constant overhead ε.

4 Experiments and results

Two series of experiments are implemented for the two different objectives as described in the previous section. In each experiment, *tags* are determined by Fibonacci numbers and the specified source symbols size k. Tags are encoded as an individual, $v(i)$, and represent that only these degrees have non-zero probabilities. Initial values of tags are set as $1/|v|$ uniformly and then CMA-ES is applied without any customization or modification. After a new individual is created, it is normalized to be a valid probability distribution and evaluated for the fitness value by simulating the LT coding process. One hundred independent runs of simulation are conducted for each function evaluation. In the first series of experiments, we minimize the expected number of encoded symbols for full decoding. In the second, the average number of source symbols that cannot be recovered for a constant $\varepsilon = 1.1$ is considered. We call the second indicator as *failure rate*. The default parameter settings given in the source code of CMA-ES are adopted in this study except for $\lambda = 10$.

4.1 Overhead

In the first set of experiments, we try to minimize the overhead ε for different k's. Fig. 2 presents the improvement throughout the generations during the evolutionary process. The initial value of an individual a uniform distribution. It is expected that overheads are quite high in the beginning and the curves descend quickly after around 100 function evaluations. Finally, the fitness almost converges after 200 function evaluations. Fig. 3 shows the comparison of ε between

Table 1: The best individuals for the optimization of overhead

Degree	k=100	$k = 400$	$k = 400$	$k = 1000$
1	0.091397	0.116375	0.16058	0.129707
2	0.310884	0.255701	0.148543	0.266133
3	0.367223	0.34174	0.412275	0.321489
5	0.042648	0.112072	0.119163	0.077045
8	0.053247	0.071726	0.052843	0.124503
13	0.048949	0.028076	0.024701	0.000258
21	0.011876	0.013169	0.035112	0.019594
34	0.073776	0.030397	0.017738	0.033607
55	0	0.000264	0.002094	0.01543
89	0	0.01109	0.009837	0.00095
144	0	0.01939	0.002946	0.000143
233	0	0	0.014167	0.00075
377	0	0	0	0.010391

the robust soliton distribution and optimized distributions. The expected overhead of robust soliton distribution is given as:

$$\frac{k + O(\ln^2(k/\delta)\sqrt{k})}{k} = 1 + O(\frac{\ln^2(k/\delta)}{\sqrt{k}}) .$$

The value becomes smaller when k increases, and that is why the trend of Fig. 3 shows a declination. The values of overhead are reduced at least 10% for all k's with the optimized degree distributions. Some distributions of the best individuals are given in Table 1. Fig. 4 illustrates each distribution and shows the histogram of successful rate in 1000 simulation runs on the right side. Compared with similar simulation results of robust soliton distribution in Fig. 5, the improvement is obvious.

4.2 Failure rate

Unlike the original LT codes, we are concerned with how many source symbols can be recovered in the second set of experiments. The objective value is to compute the average number of source symbols that cannot be recovered with a constant overhead ε. Optimization results are shown in Fig. 6. More function evaluations are needed to search for good degree distributions. The failure rate of the final results are less than 10^{-1} for all k's when $\varepsilon = 1.1$. In other words, 90 percent of source symbols can be successfully recovered if extra 10 percent encoded symbols are collected. Table 2 gives the best probability distributions found in the evolutionary process for $k = 100$, $k = 400$, $k = 700$, and $k = 1000$. The simulation results of a constant overhead are presented in Fig. 7. The red line denotes the behavior of uniform distribution which is the initial value of optimization. Most of the source symbols are un-recovered except for those of which the degree is one, i.e., with probability $1/k$. The same situation happens to robust soliton distributions because extra encoded symbols are not sufficient to complete the BP decoding process. The behavior of LT process with the optimized degree distributions is totally different and satisfies the characteristics of weakened LT.

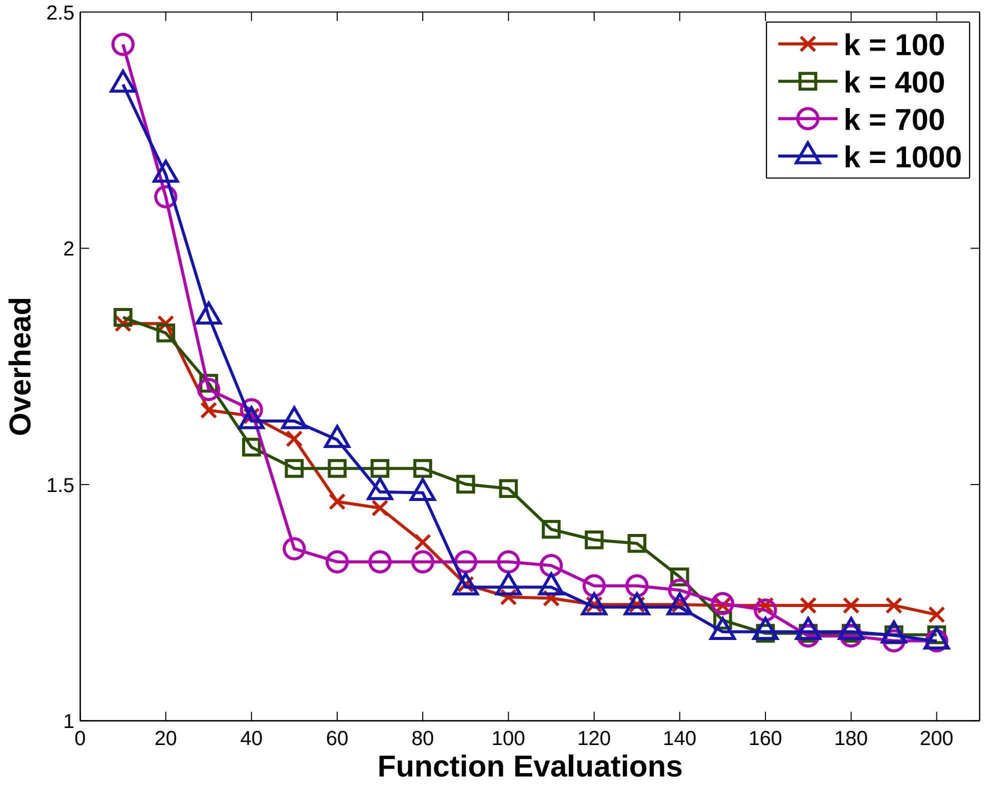


Figure 2: Evolutionary process during the optimization of overhead

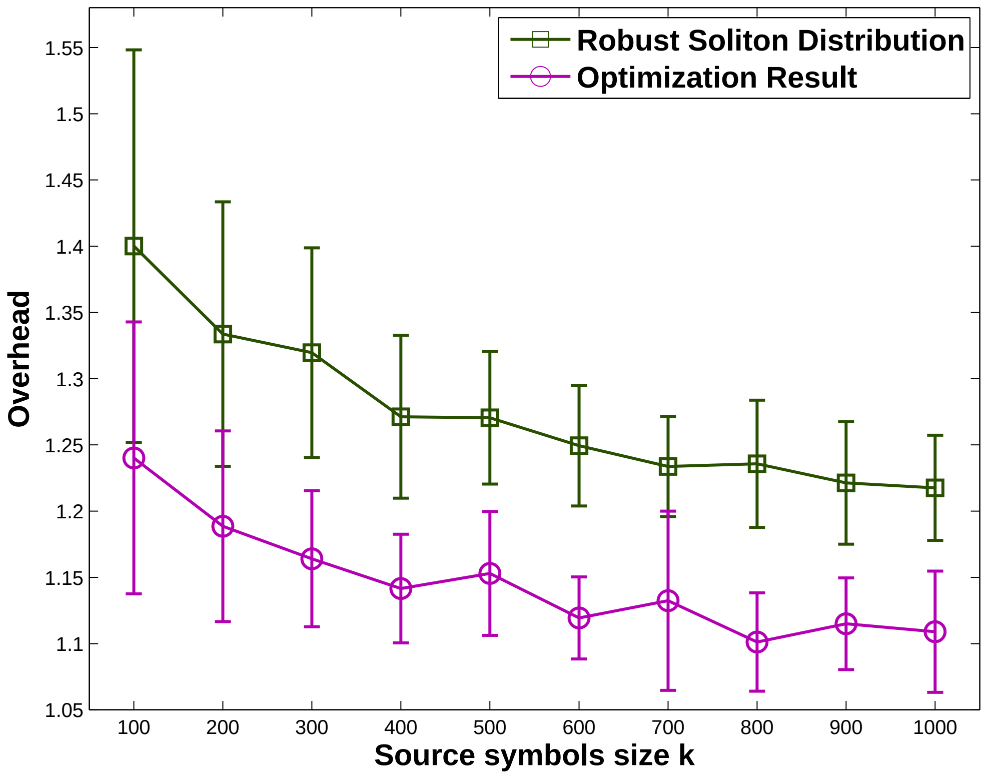
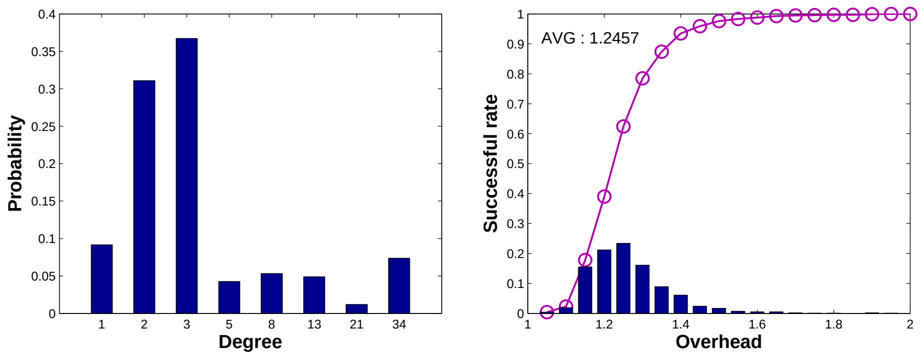
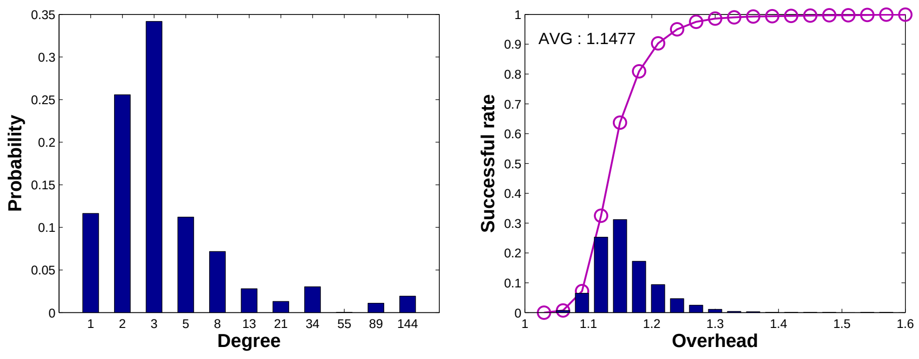


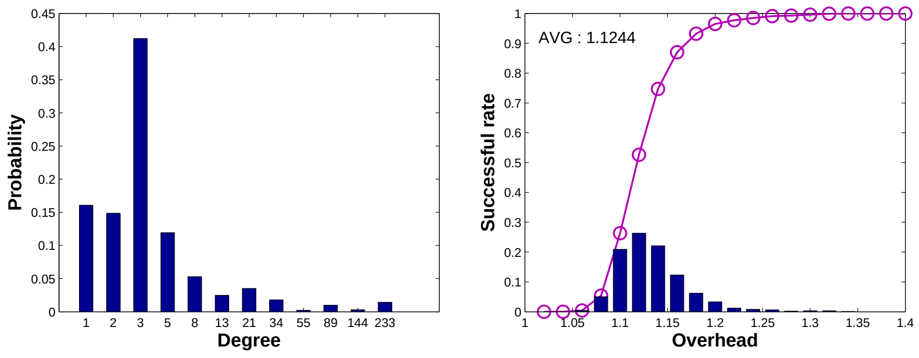
Figure 3: Average performance indicators are compared between robust soliton distribution and optimized degree distributions for different numbers of source symbols (k)



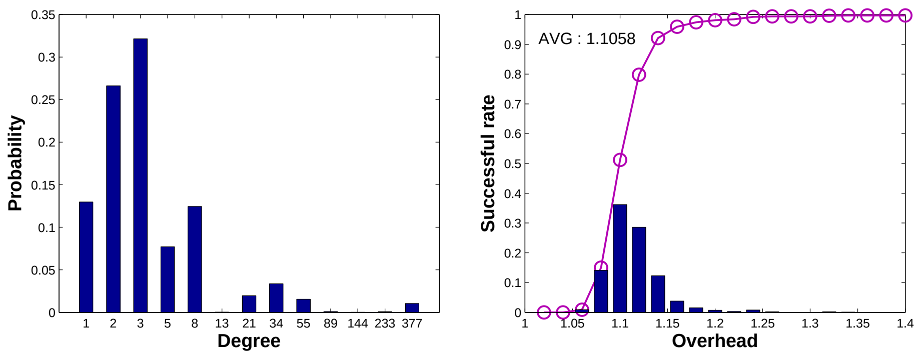
(a) $k = 100$



(b) $k = 400$

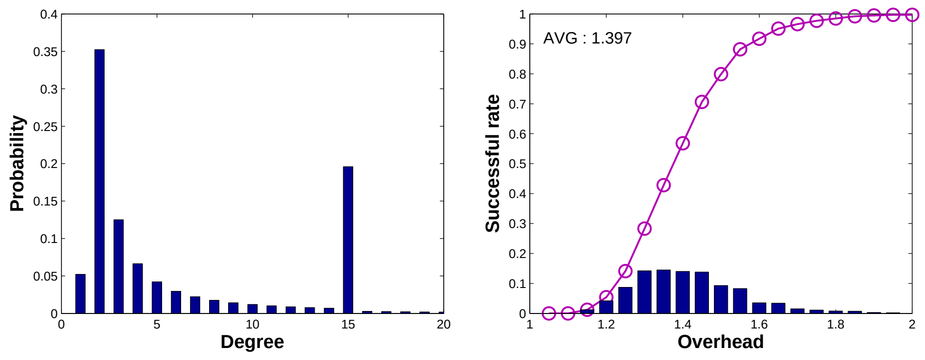


(c) $k = 700$

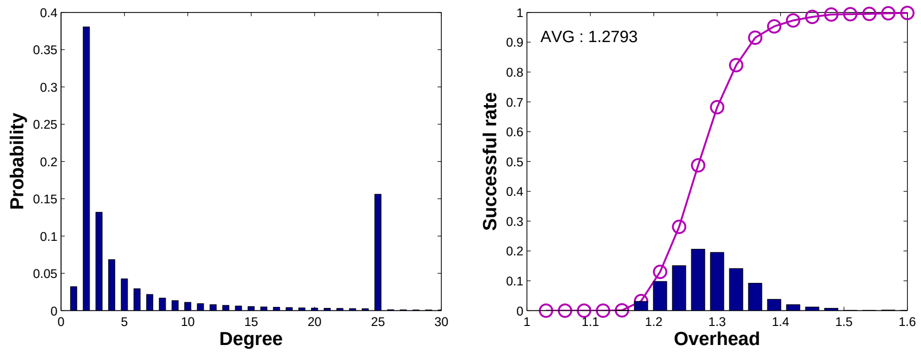


(d) $k = 1000$

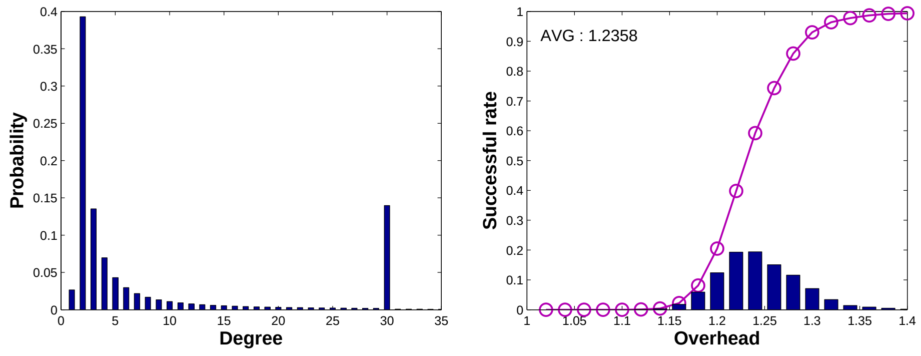
Figure 4: Left figures show the optimized degree distributions. Only tags are presented. Right figures are the histogram and accumulated curve of successful rate in 1000 independent simulation runs



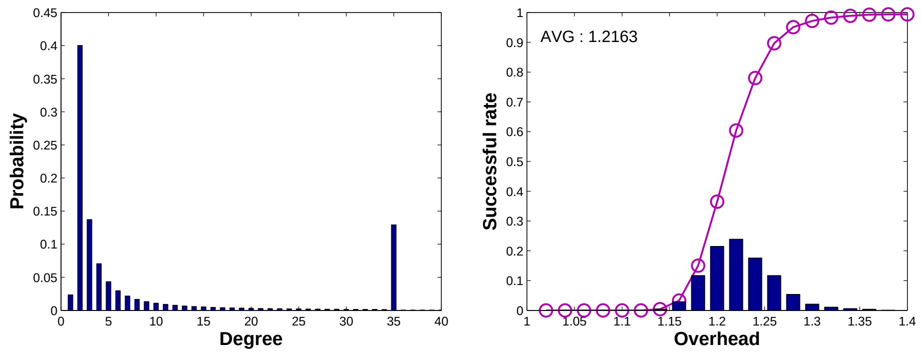
(a) $k = 100$



(b) $k = 400$



(c) $k = 700$



(d) $k = 1000$

Figure 5: For the comparison with same k's, robust soliton distributions and the corresponding performance indicators are shown similar to that in Fig. 4. Note that only parts of robust soliton distributions are plotted for clarity

Table 2: The best individuals for the optimization of failure rate

Degree	k=100	k = 400	k = 400	k = 1000
1	0.083997	0.102892	0.116854	0.115278
2	0.573671	0.383164	0.29678	0.333564
3	0.161178	0.237312	0.31115	0.241065
5	0.08038	0.186475	0.171342	0.184027
8	0.096245	0.030706	0.033393	0.046818
13	0.001267	0.039075	0.025977	0.022223
21	0.002963	0.015193	0.023452	0.022914
34	0.000299	0.000167	0.016096	0.020526
55	0	0.001276	0.002602	0.00643
89	0	0.000303	0.000268	0.004594
144	0	0.003436	0.002072	0.001422
233	0	0	0.000015	0.000883
377	0	0	0	0.000257

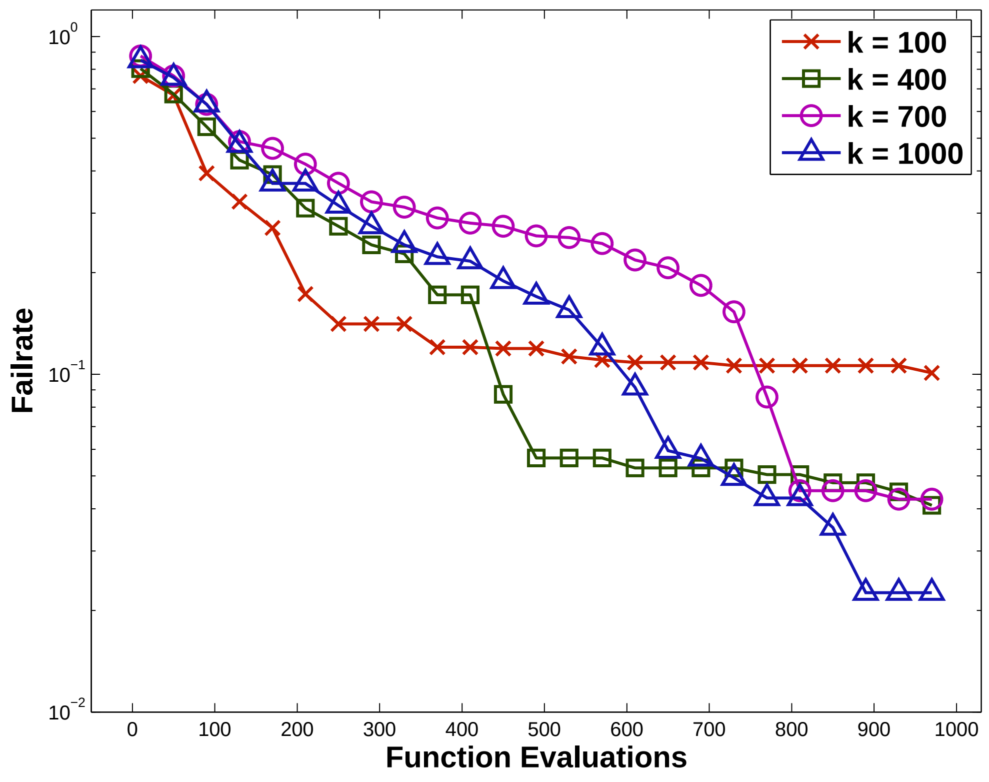
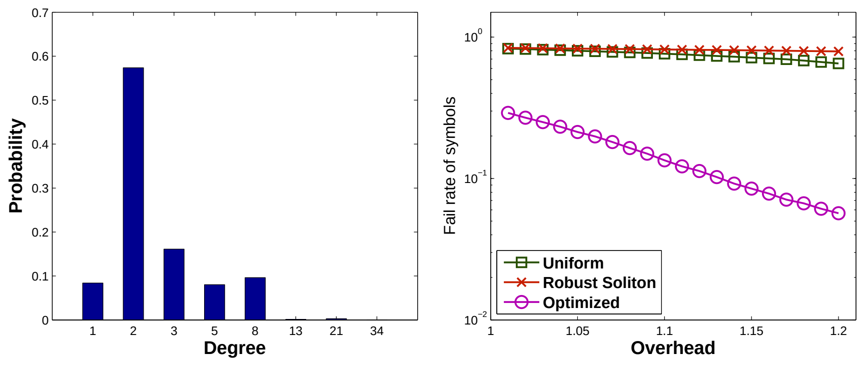


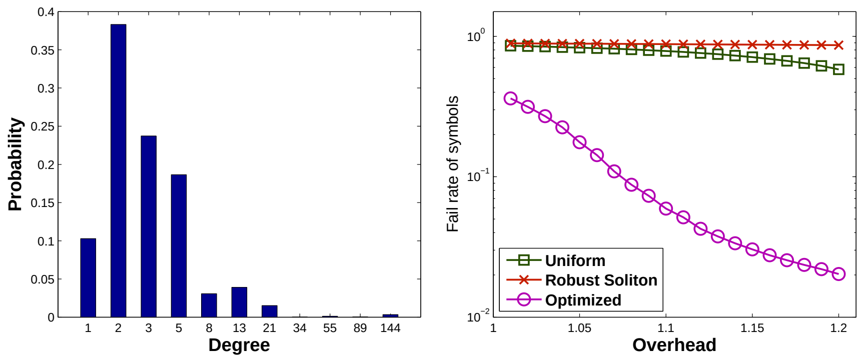
Figure 6: Evolutionary process during the optimization of failure rate

5 Conclusions

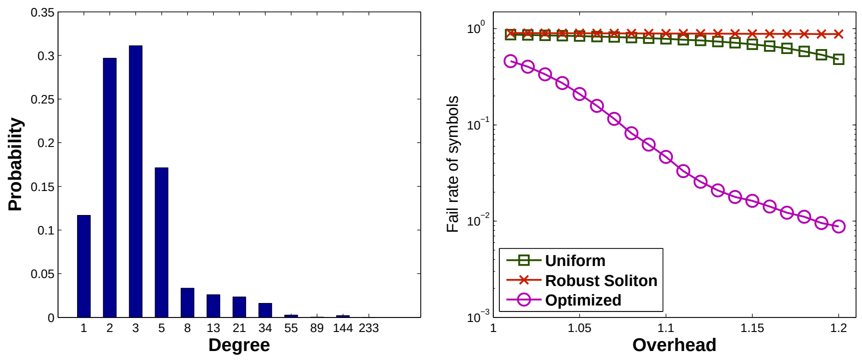
In this work, the first attempt to algorithmically optimize the degree distribution adopted in LT codes was proposed. Evolutionary computation techniques were introduced to accomplish the optimization task. Different from the previous studies reported in the literature, each probability of degrees were directly encoded as an individual to optimize. Promising experimental results were obtained in both sets of experiments: One was to minimize the overhead, and the



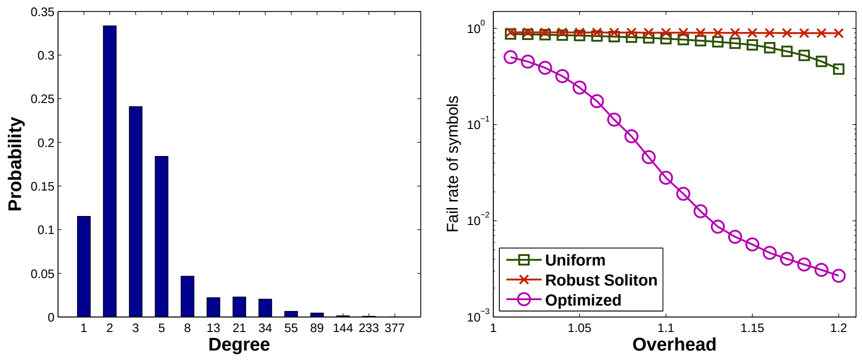
(a) $k = 100$



(b) $k = 400$



(c) $k = 700$



(d) $k = 1000$

Figure 7: The figure shows the significant difference of failure rate after optimization. Similar to that in Fig. 4, only tags are shown in the figures

other was to reducing the decoding failure rate. Our experiments showed that CMA-ES was indeed capable of finding good degree distributions for different purposes without any guideline or human intervention. Compared with the robust soliton distribution, the optimized overhead was decreased as least 10% for every k in the experiments. The results of failure rate minimization were also remarkably promising and able to support applications of different types and requirements.

This study creates a new research topic in which the design of degree distributions in LT codes can now be algorithmic and no longer has to be manually tuning parameters of robust soliton distribution. We have empirically proved that directly manipulating the probability value for each degree is viable and worth pursuing. Given a specific k and some expected overhead, a degree distribution can be customized with existing optimization techniques. In addition, we will extend the experiments to larger k for more kinds of potential applications in the near future. The results empirically obtained by using evolutionary algorithms will be theoretically analyzed, and general guidelines, like robust soliton distribution, that are able to be customized for different goals and requirements for designing degree distributions are 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] O. Etesami, M. Molkaraie, and A. Shokrollahi, "Raptor codes on symmetric channels," in *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, 2004, p. 38.

[8] O. Etesami and A. Shokrollahi, "Raptor codes on binary memoryless symmetric channels," *Information Theory, IEEE Transactions on*, vol. 52, no. 5, pp. 2033–2051, 2006.

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

[10] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation," in *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, 1996, pp. 312–317.

[11] 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.

[12] A. Auger and N. Hansen, "Performance evaluation of an advanced local search evolutionary algorithm," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 2, 2005, pp. 1777–1784 Vol. 2.

[13] ——, "A restart cma evolution strategy with increasing population size," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 2, 2005, pp. 1769–1776 Vol. 2.

[14] R. Ros and N. Hansen, "A simple modification in cma-es achieving linear time and space complexity," *PPSN*, pp. 296–305, 2008.

[15] R. Gallager, "Low-density parity-check codes," *Information Theory, IRE Transactions on*, vol. 8, no. 1, pp. 21–28, 1962.

[16] D. Changyan, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *Information Theory, IEEE Transactions on*, vol. 48, no. 6, pp. 1570–1579, 2002.

[17] E. Paolini and M. Chiani, "Improved low-density parity-check codes for burst erasure channels," in *Communications, 2006. ICC '06. IEEE International Conference on*, vol. 3, 2006, pp. 1183–1188.