Design of Optimal Short-Length LT Codes Using Evolution Strategies

John K. Zao*, Martin Hornansky, Pei-lun Diao

Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, R.O.C.
e-mail: jkzao@cs.nctu.edu.tw, martin.hornansky@gmail.com, klandor@gmail.com

Abstract—Luby Transform (LT) and its companion Raptor codes are the most popular implementations of *digital fountain codes*. Performance of these rateless forward erasure correction codes is determined mainly by the *degree distributions* of their encoded symbols. Although the asymptotic behaviors of LT codes with large ($>10^5$) symbol blocks have been deduced analytically, a proficient method for finding the optimal degree distributions of short length ($<10^3$) LT codes is still absent. In this paper, we propose a practical approach to employ evolution strategies in finding the degree distributions of optimal short-length LT codes for different applications. Our approach begins with the development of a new performance model for LT codes based on three measurements: coding overhead ε , failure ratio r and failure occurrence probability p . Three evolution strategies (DE, CMA-ES and NES) were then employed to minimize these performance measurements separately with careful design of fitness functions and necessary transformations of decision variables. Throughout the evolution process, the performance of individual LT code in the population was evaluated with numerical simulations. Our experiments showed that the optimal degree distributions can be found using all three evolution strategies but with different convergence rates and the (r, p, ε) values of these optimized codes are all distributed on a smooth concave surface.

Keywords — *Digital Fountain Codes, Luby Transform (LT), Differential Evolution (DE), Covariance Matrix Adaptation - Evolution Strategy (CMA-ES), Natural Evolution Strategy (NES)*

I. INTRODUCTION

DIGITAL FOUNTAIN CODES [1][2] were introduced by Byers et al. in 1998 [3] as a new class of forward erasure correction (FEC) codes [4] for protecting asynchronous data transfer over packetized digital networks. These rateless codes can produce a limitless supply of randomly encoded symbols from given source symbols. The encoded symbols can then be put into data packets and sent via multicasting or broadcasting. Receivers over the networks can simply capture any packets in the data streams at any time. Once sufficient number of packets is captured—which usually equal to the amount of source symbols plus a small overhead—a receiver can decode the received packets and most likely recover all the source symbols using a *belief propagation* algorithm with linear computation complexity. If the receiver fails to recover some of the source symbols, it may capture a few more packets and continue the decoding process until all the symbols are recovered. The scenario is like filling a cup from a water fountain and hence the name of these codes.

Luby [5] proposed the first practical implementation of digital fountain codes in 2002 and christened them, the *Luby Transform (LT) codes*. The decoding failure rate of an LT code

is completely determined by the amount of source symbols K it handles, the *degree distribution* $\rho(d)$ it uses to conduct the randomized encoding process and the amount of *encoding symbols* N it receives before starting the decoding process. In order to improve the performance of an LT code, Shokrollahi added a low-density parity check (LDPC) code to the LT codes as a precoder and called these composite codes, the *raptor codes* [6]. Due to their rateless forward erasure correcting capability and Shannon capacity approaching coding efficiency the raptor codes become widely used in data transfers over wireless networks and the Internet.

Since their inception, the asymptotic behaviors of LT codes with infinite amount of source symbols have been deduced analytically. However, the deduction cannot be extended to the cases with small number of source symbols as the assumption of ergodic behaviors is no longer valid. Over the past years, researchers made various attempts to establish design guidelines for optimal “short length” LT codes or *SL-LT codes*. Those attempts [8]–[11] were met with limited success. Thus, the design of optimal SL-LT codes remains an open problem.

In this paper, we propose a practical and robust approach to employ *evolution strategies* in designing optimal SL-LT codes with decoding performance that suit different applications. The success of our approach depends largely on a new performance model we devised for the SL-LT codes in terms of their coding overhead ε , failure ratio r and failure occurrence probability p . Special efforts have also been made to ensure proper use of the evolution strategies. They include (1) the selection of evolution strategies by balancing their convergence rates and computation efficiency, (2) the choice of *decision variables*, (3) the specification of *fitness functions*, (4) the choice of *initial population* and (5) the *criteria* for selecting population samples in every generation. Our experiment outcomes were remarkably promising: significant reduction of overheads, failure rates and failure probabilities have been achieved for SL-LT codes with various block sizes.

The remainder of this paper is organized as follows. Section II provides a brief overview of LT encoding/decoding operations including the soliton distributions. Section III describes (r, p, ε) based LT code performance model. Section IV formulates the optimization problem and describes its two scenarios. Section V discusses the rationale for choosing the three evolution strategies while Section VI specifies the designed initial distributions, transformation of decision variables and the fitness penalty added to the out-of-range samples. Experiment results and their analyses are given in section VII and contributions are summarized in Section VIII.

This research was supported in part by the ATU Program Phase II under the Taiwan Ministry of Education [Grant N.: 101W963] and the UST-UCSD International Center of Excellence in Advance Bio-engineering under the Taiwan National Science Council I-RiCE Program [Grant No.: NSC-100-2911-I-009-101].

II. LT CODES

A. Encoding and Decoding Operations

In binary LT codes, each source symbol is a binary tuple. These source symbols are randomly chosen and combined into an *encoded symbol* or *codeword* by a *bitwise exclusive OR* (XOR) operation. The number of source symbols combined into an encoded symbol is known as its *degree*. The relation between the source and encoded symbols can be represented as a bipartite graph or a binary encoding matrix with each column showing the presence or absence of connections between an encoded symbol and the source symbols.

A block of K source symbols can be decoded or recovered from the same amount or more encoded symbols with the use of *Gaussian elimination*. However, this maximum likelihood algorithm with $\mathcal{O}(K^3)$ complexity can be replaced by a sub-optimal *belief propagation* (BP) algorithm [12] with only $\mathcal{O}(K)$ complexity.

Figure 3 illustrates the BP decoding process with a simple example. The only price for using this simpler process would be a small increase in the number of encoded symbols necessary to ensure successful decoding and the devise of “good” degree distributions to work with the BP algorithm.

B. Degree Distribution

The degree distribution $\rho(d)$ of an LT code is a discrete probability mass function that specifies the probability ρ of producing an encoded symbol from d source symbols [13]. In his LT code proposal [5], Luby deduced an *ideal soliton* degree distribution based on an analogy of throwing infinite number of balls randomly into K number of bins:

$$\begin{aligned} \rho(1) &= \frac{1}{K} \\ \rho(d) &= \frac{1}{d(d-1)} \quad \text{for } 1 < d \leq K \end{aligned} \quad (1)$$

With its theoretical rigor, ideal soliton is however not suitable for practical use because the decoding behavior of finite-length LT codes may fluctuate among randomly generated *codewords*. In the same paper, Luby proposed another degree distribution, known as the *robust soliton* distribution. Its probability mass function $\mu(d)$ can be derived by adding an adjustment term $\tau(d)$ to that of the ideal soliton distribution:

$$\mu(d) = \frac{\rho(d) + \tau(d)}{\sum_{d=1}^K \rho(d) + \tau(d)} \quad (2)$$

$$\tau(d) = \begin{cases} \frac{S}{K} \left(\frac{1}{d} \right) & \text{for } d \in [1, 2, \dots, (S/R) - 1] \\ \frac{S}{K} \ln \left(\frac{S}{\delta} \right) & \text{for } d = S/R \\ 0 & \text{for } d > (S/R) \end{cases} \quad (3)$$

Where K is the number of source symbols and δ is the failure probability of decoding process for some real constant $c > 0$.

Robust soliton is good for practical use but only in the cases with infinitely many source symbols. Even though Karp et al. later published a recursive formula for estimating performance of finite-length LT codes in [7], no formula is currently

Figure 1: LT code decoding process

available for constructing optimal degree distributions for finite-length LT codes not to mention those for SL-LT codes.

C. Code Performance

The performance of finite-length LT codes has been hard to predict because these codes employ a randomized encoding process. Unlike the fixed-rate deterministic codes, which offer sturdy erasure correction capability until the symbol loss rate exceeds a limit, the randomized encoding process of LT code may produce a sequence of highly decodable codewords in one instance and a sequence of poorly decodable ones in another. Figure 2 shows the “bi-modal” profile of the decoding failure rates of a SL-LT code with $K = 1000$. There is a high chance that up to 70% of source symbols may not be recovered if only a small number of codewords were used for decoding. In these situations, the conventional approaches of minimizing the average failure rate or the total area under the failure probability profile may be misleading. The proper objective

Figure 2: Histogram of decoding failure rates of an LT code with $K = 1000$. The number of codeword is equal to $K(1 + \epsilon)$

should be the reduction of the maximum failure rate or the probability of high failure instances.

D. Code Applications

Another reason for devising a new model to quantify the performance of finite-length LT codes is that different applications want the performance of the codes to be optimized in different fashions. To our knowledge, there are at least three distinct applications of finite-length LT codes.

1) *Erasur protection for lossless data transfer*: Applications such as file downloads demand perfect data reception among the receivers. These applications also prefer to transmit as few symbols as possible in order to avoid waste of communication bandwidth or increase in transport latency. These applications may also conduct data transfers in asynchronous receiver-driven fashion so that a receiver can obtain data from several broadcasting/multicasting sources.

2) *Data transfer with limited overhead allowance*: Applications such as video streaming can tolerate small amount of decoding failures as long as the pictures can be playback with reasonable quality. These real-time applications however can not tolerate large increase in bandwidth or latency. In other words, they expect to have the best picture quality (or lowest error rates) attainable under available data rates.

3) *Postcoding in rateless composite codes*: In these cases, the LT codes only need to bring the decoding failure rates below certain threshold so that the precoder can recover the missing data symbols. Hence, the LT codes can tolerate certain amount of decoding failures but expect the failure rates to remain almost always below the threshold level.

III. PERFORMANCE MODELS

Based on our understanding towards the “bi-modal” performance profile and the different uses of finite-length LT codes, we devised a new performance model for these codes in terms of three basic measurements:

- ❖ *Overhead* $\{\varepsilon \mid \varepsilon \geq 0\}$: It is the ratio between the number of extra encoded symbols received in comparison with the number of source symbols.
- ❖ *Failure Rate* $\{r \mid 0 \leq r < 1\}$: It is the fraction of the unrecovered source symbols during a decoding process.
- ❖ *Failure Probability* $p = \Pr(R_\varepsilon > r)$: It is the probability of the decoding failure rate to be higher than a threshold value r while the code is decoded with an overhead ε .

Each type of applications mentioned above would like to use LT codes that are optimized with respect to one of these three measurements while having the other two maintained within their acceptable ranges. Obviously, lossless data transfers want to minimize the overhead ε ; multimedia streaming want to reduce their transfer failure rate r while the post-coders should always minimize their decoding failure probability p .

IV. OPTIMIZATION PROBLEM

A. Design Variables

Because our ultimate goal is to find the best degree distributions using evolutionary strategies, we shall begin with the design of the *decision variables* that can properly represent the probability mass function $\rho(d)$. For that purpose, we created two M -tuples: the *degree tuple* $\mathbf{d} \stackrel{\text{def}}{=} [d_i]_{i=1}^M$ and the *probability tuple* $\mathbf{p} \stackrel{\text{def}}{=} [p_i]_{i=1}^M$. In the two tuples, each pair of corresponding elements (d_i, p_i) captures a *non-trivial entry* of the probability mass function:

$$\forall i \in [1, 2, \dots, M] \quad p_i \triangleq \rho(d_i) \text{ iff } \rho(d_i) > 0 \quad (4)$$

Since the degree distribution of SL-LT codes is sparse for most actual applications, the dimension M of the tuples can be set at a much lower value than the maximum degree K .

We also encountered a few practical issues when these tuples were employed in the actual optimization processes: (1) Degree elements need to be rounded to the closest integers within $[1, K]$. (2) Several degree elements $d_{i \in \Gamma}$ may arrive at the same value in the final result. In those cases, we need to sum up *all* the corresponding probability elements $p_{i \in \Gamma}$ in order to obtain the actual probability value $\rho(d_i)$. (3) Since a fixed dimension M was chosen for each optimization process, some probability elements $p_{i \in \Xi}$ may become insignificant in the final results. In those cases, these probability values may be added to the significant probability element of the closest degree(s).

B. Fitness Function Evaluation

Although Karp et al. derived a recursive formula for estimating the decoding failure rates of finite-length LT codes [7], the use of this formula is only computationally feasible if the number of source symbols remains very small ($K \sim 10^2$). Hence, we decided to evaluate the performance of different SL-LT code samples in each generation by means of numerical simulation of the actual decoding process. In order to speed up the computation, we used incremental BP decoding and stop the simulation once approx. 10^2 failures were cumulated at every test point.

C. Optimization Scenarios

Since we were unsure of the convergence behaviors of designed optimization scheme, we devised two scenarios to test how the evolution strategies would behave in different settings.

1) *Fixed Degree Scenario*: In this scenario, the components of the degree tuple are kept constant throughout the optimization process. Thus, the design variables consist only of the components of probability tuple.

2) *Variable Degree Scenario*: In this scenario, the design variables consist of the components of both the degree and the probability tuples. In other words, the design variables become *heterogeneous* in nature: some of them represent the degrees of encoding symbols while the others specify the probability of having those degrees.

V. OPTIMIZATION METHODS

We attempted to employ a new performance model along with three different evolution strategies to search for SL-LT codes with specific performance characteristics.

A. Covariance Matrix Adaptation Evolution Strategy

CMA-ES [14] is the de-facto evolutionary optimization strategy. It works by iteratively updating the covariance matrix of a multivariate normal distribution of mutated population. CMA-ES is especially successful in solving badly conditioned, multimodal and noisy problems with high-dimensional rugged search landscape. We chose CMA-ES as a benchmark method for our experiments.

B. Natural Evolution Strategy

NES [15] is a new numerical optimization method, which performs gradient ascent along the natural gradient in the population parameter space [16]. The employment of natural gradient helps to prevent oscillations, premature convergence, and other undesired effects. Akitomo et al. [17] showed that CMA-ES can be regarded as a special case of NES if its step-size control value c_σ and rank-one update value c_1 are set to zero. In our experiments we used an enhanced *exponential NES* (*xNES*) [18] for its more robust convergence behavior.

C. Differential Evolution

DE [19] is an evolution strategy, which creates new candidate solutions by combining existing ones according to simple formulae based on vector differences. DE performs especially well on noisy constrained optimization problems with multidimensional real-valued functions and problems that change over time. In our experiments we have obtained consistent and robust results applying DE versions *Rand/1/Bin* and *Rand/2/Bin*, while versions *Rand/1/Exp* and *Rand/2/Exp* as well as all corresponding *Best* versions in most cases suffered from premature convergence.

VI. OPTIMIZATION SCHEME

Various issues needed to be considered as we employed different evolution strategies (ES) for the design of optimal SL-LT codes with specific performance characteristics.

A. Decision Variables

1) *Transformations between Decision and Design Variables:* Since the *design variables* of SL-LT codes, i.e. the degree tuple $\mathbf{d}$ and the probability tuple $\mathbf{p}$, have bounded value ranges, it is necessary to adapt their ranges to those of the *decision variables* of the evolution strategies. Towards that end, we defined two decision variable tuples: $\mathbf{x} \triangleq [x_i]_{i=1}^M$ and $\mathbf{y} \triangleq [y_i]_{i=1}^M$ as well as the necessary transformations between the decision and the design variables: $\mathbf{x} \rightarrow \mathbf{d}$ and $\mathbf{y} \rightarrow \mathbf{p}$. ES population samples are specified in terms of decision variables, while the fitness functions are evaluated based on design variables. Separate transformations were devised for different evolution strategies. This is because CMA-ES and NES are stochastic strategies that use Gaussian distributions to produce random off-springs over an unbounded variable space

while DE is a population-based evolution strategy that can evolve its off-springs in a bounded variable space.

- ❖ *Probability Transformation for CMA-ES and NES:* A quadratic transformation is used to map the tuple of unbounded decision variables $\mathbf{y} \triangleq [y_i]_{i=1}^M$ to the tuple of probability variables, $\mathbf{p} \triangleq [p]_{i=1}^M$:

$$p_i = \frac{y_i^2}{\sum_{j=1}^M y_j^2} \quad (5)$$

- ❖ *Probability Normalization for DE:* Because DE is population based linear optimization method, we only need to perform following normalization:

$$p_i = \frac{|y_i|}{\sum_{j=1}^M |y_j|} \quad (6)$$

- ❖ *Degree Rounding and Constraining:* CMA-ES, NES and DE can during the evolution process generate samples with degree decision variables $\mathbf{x}$ being real numbers and lying beyond the feasible range of corresponding degrees. We map them to the design variables $\mathbf{d}$ as follows:

$$d_i = \begin{cases} 1 & \text{for } x_i < 1 \\ \text{round}(x_i) & \text{for } 1 \leq x_i \leq K \\ K & \text{for } x_i > K \end{cases} \quad (7)$$

- ❖ *Decision Variables Adjustment for DE:* Throughout the evolution process, DE can generate samples with the decision variables $\mathbf{x}$ and $\mathbf{y}$ being located beyond the feasible range of $\mathbf{d}$ and $\mathbf{p}$. However in DE, only legitimate samples can generate a new offspring. Therefore it is necessary to perform an appropriate adjustment to ensure that every sample has legal values of decision variables. In our implementation, we first calculate valid design variable values as in equations (5) and (7), and then copy them back to corresponding decision variables as follows:

$$\forall i \in [1, 2, \dots, M] \quad x_i = d_i \wedge y_i = p_i \quad (8)$$

2) *Penalty for Out-of-range Degree Values:* The fact that CMA-ES and NES can generate unbounded values for their variables affects both, degrees and probabilities. A quadratic function was thus introduced to add monotonically increasing penalty to the fitness value when the degree values lie beyond their admissible range:

$$\text{Penalty}(\mathbf{x}) = \sum_{i=1}^M [x_i - K] + \sum_{i=1}^M [x_i - 1] \quad (9)$$

$$\text{Where } [x_i - K] = \begin{cases} (x_i - K)^2 & \text{for } x_i > K \\ 0 & \text{for } x_i \leq K \end{cases} \quad (10)$$

$$[x_i - 1] = \begin{cases} (x_i - 1)^2 & \text{for } x_i < 1 \\ 0 & \text{for } x_i \geq 1 \end{cases}$$

B. Initial Degree Distribution

Both CMA-ES and NES require their users to specify the initial values of their decision variables. For the optimization of SL-LT codes, these initial values correspond to their initial degree distribution.

1) *Choices of Initial Degrees*: Since the SL-LT codes often have small average degrees and sparse degree distributions, we chose their initial degrees from the range of $[1, K/5]$. As shown in TABLE I, we chose the degree values to be either the prime numbers or the powers of two in the fixed degree optimization scenario so that they are spread even over a logarithmic scale and do not possess a common factor. The initial degree values used in the variable degree scenario are underlined elements of TABLE I. They were the ones having the highest probabilities in the fixed degree scenario.

TABLE I. INITIAL DEGREE VALUES FOR FIXED DEGREE SCENARIO

<u>1</u>	<u>2</u>	<u>3</u>	4	5	<u>7</u>	8	11	16	19
<u>23</u>	32	<u>41</u>	53	64	83	101	128	163	<u>199</u>

2) *Specification of Initial Probabilities*: We started our optimization experiments using three different initial degree distributions. Two of them were derived from the soliton distributions. We chose them because ideal and robust soliton work perfectly well in the asymptotic cases of infinite-length LT codes. We chose *uniform distribution* as the third alternative so that we can perform a “stress test” onto the optimization strategies with initial values located far from the optimal solution.

❖ *Sparse Robust Soliton Distribution*: This sparse initial degree distribution was created by gathering probability values under the adjacent degrees of the robust soliton distribution to those under the selected initial degrees. The boundaries $\{\beta_i\}$ of these gathering intervals are set at the midpoints between two adjacent initial degrees:

$$p_i = \sum_{d \in [\beta_i, \beta_{i+1})} \mu(d)$$

$$\beta_i = \begin{cases} 1 & \text{for } i = 1 \\ \left\lceil \frac{(d_{i-1} + d_i)}{2} \right\rceil & \text{for } i \in [2, 3, \dots, M] \\ K + 1 & \text{for } i = M + 1 \end{cases} \quad (11)$$

Where $\mu(d)$ is the probability mass function of robust soliton distribution and the degree tuple is sorted in an increasing order.

❖ *Sampled Ideal Soliton Distribution*: Except the last one, every component of the probability tuple was assigned the corresponding value of the ideal soliton distribution. The last component p_M then absorbed the remaining probability [20]:

$$p_i = \begin{cases} \rho(d_i) & \text{for } i \in [1, 2, \dots, M - 1] \\ 1 - \sum_{j=1}^{M-1} p_j & \text{for } i = M \end{cases} \quad (12)$$

❖ *Uniform Distribution*: Every component of the probability tuple was assigned the same value:

$$p_i = \frac{1}{M} \quad \text{for } i \in [1, 2, \dots, M] \quad (13)$$

Figure 3. ϵ - r - p space displaying 56 optimization results

VII. EXPERIMENTS AND RESULTS

A. Specific Values of Performance Measurements

To test the optimization strategies, we tried to find optimal SL-LT codes based on the following performance measurements that may appear in real-world situations:

TABLE II. PERFORMANCE MEASUREMENT VALUES USED IN EXPERIMENTS

ϵ	0.03, 0.06, 0.1, 0.13, 0.16
r	0, 10^{-3} , 10^{-2} , 10^{-1}
p	10^{-4} , 10^{-3} , 10^{-2} , 10^{-1}

In each optimization task, we fixed the values of two performance measurements while optimizing the third one.

B. Parameters of Optimization Methods

1) *CMA-ES and NES Cases*: Users only need to specify the initial decision variable values and their standard deviations. For the standard deviations, we chose the values from $[10, 30]$ and $[0.02, 0.2]$ for decision variables representing degrees and probabilities respectively, depending on the kind of the initial degree distribution being used.

2) *DE Parameters*: In our experiments, we assigned the values of crossover probability and scaling factor to be equal 0.5 and 0.7 respectively, as suggested [19][21][22] to be used when the optimization problem is continuous, noisy, and exhibit respective problem dimensionality.

C. Comparison of Convergence Behaviors

We traced the fitness function values and average degrees of every generation throughout the evolution process and plotted the behaviors of different optimization runs in Figure 4 to Figure 12. Among them, Figure 4–Figure 6 show the results of the experiments at which each of the three performance measurements (r, p, ϵ) were being optimized. Convergence behaviors of CMA-ES/NES/DE are compared in Figure 7 – Figure 9 while the convergence behaviors of CMA-ES with

three different initial degree distributions are shown in Figure 10–Figure 12.

D. Consistency among Optimized Performance

Figure 3 shows the optimized performance measurements obtained using CMA-ES as coordinates in the (r, p, ε) space. The blue, purple and red dots highlight the results that minimized overhead ε , failure rate r and failure probability p respectively. Obviously, these dots are distributed over a concave surface. We regard this phenomenon as evidence that CMA-ES has found the truly optimized values. Also, certain basic properties of the LT codes are dictating their performance.

E. Observations

We can summarize the results of our experiments into the following observations.

- ❖ Evolution strategies were shown to be a practical method for designing optimal SL-LT codes. All three strategies used in our experiments managed to converge and produce degree distributions for SL-LT codes that perform much better than the robust soliton distribution.
- ❖ DE appeared to be inferior to CMA-ES and NES in terms of the rate and robustness of its convergence behaviors.
- ❖ CMA-ES and NES showed similar convergence behaviors regardless of the choice of initial degree distribution.
- ❖ NES appeared to be the most robust evolutionary strategy for finding the optimal SL-LT codes. In a few stringent cases, NES succeeded in producing optimal results while CMA-ES has failed to converge.
- ❖ The optimization scenarios with fixed and variable degrees produce similar performance measurement values as well as similar degree distribution profiles although the variable degree scenario tends to produce more sparse distributions.

VIII. CONCLUSION

In this paper, we propose a practical approach to employ evolution strategies (ES) in finding the degree distributions of optimal short-length LT codes. Experiment results confirmed that ES, especially NES, can produce optimal degree distributions suitable for different applications. Moreover, we believe that the consistent results obtained by applying three different ES to different optimization tasks with different initial degree distributions is a sign that certain basic properties of the SL-LT codes are dictating the performance of these codes.

ACKNOWLEDGMENT

The authors would like to thank Prof. Chung-Hsuan Wang of the NCTU Electrical Engineering Department for his valuable advices and fruitful discussion on SL-LT codes. We are also grateful to Dr. Tobias Glasmachers of the University of Bochum Institute for Neural Computation for his helpful comments on the implementation of the xNES algorithm.

REFERENCES

- [1] D. J. C. MacKay, "Fountain codes," in IEE Seminar on Sparse-Graph Codes, 2004, pp. 1–8.
- [2] D. J. C. MacKay, "Information Theory, Inference and Learning Algorithms", Cambridge University Press, 2003.
- [3] 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, 1998, pp. 56–67.
- [4] C. Wang, D. Sklar, D. Johnson (Winter 2001/2002). "Forward Error-Correction Coding". Crosslink — Aerospace Corporation magazine of advances in aerospace technology (Aerospace Corporation) 3(1). "How Forward Error-Correcting Codes Work".
- [5] M. Luby, "LT codes," in Proceedings of the Symposium on Foundations of Computer Science, IEEE Comp. Society, 2002.
- [6] A. Shokrollahi, "Raptor codes," IEEE Transactions on Information Theory, vol. 52, no. 6, pp. 2551–2567, 2006.
- [7] R. Karp, M. Luby, A. Shokrollahi, "Finite length analysis of LT codes," in Proceedings of the IEEE ISIT 2004, 2004, p. 39.
- [8] E. Hyttia, T. Tirronen, and J. Virtamo, "Optimal degree distribution for LT codes with small message length," in the 26th IEEE INFOCOM 2007, pp. 2576–2580.
- [9] E. A. Bodine and M. K. Cheng, "Characterization of Luby transform codes with small message size for low-latency decoding," IEEE International Conference in Communications, 2008.
- [10] C.-M. Chen, Y.-p. Chen, T.-C. Shen, J. K. Zao, "On the optimization of degree distributions in LT codes with covariance matrix adaptation evolution strategy," IEEE CEC, 2010.
- [11] E. Hyttia, T. Tirronen, and J. Virtamo, "Optimizing the degree distribution of LT codes with an importance sampling approach", Intl. Workshop on Rare Event Simulation, 2006.
- [12] J. Pearl, "Reverend bayes on inference engines: A distributed hierarchical approach," in Proceedings of the AAAI National Conference on AI, 1982, pp. 133–136.
- [13] T. Tirronen, "Optimizing the degree distribution of LT codes", M.S. thesis, Dept. of Electrical and Communications Eng., Helsinki University of Technology, Helsinki, Finland, 2006.
- [14] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation," IEEE CEC, 1996, pp. 312–317.
- [15] D. Wierstra, T. Schaul, J. Peters and J. Schmidhuber, Natural evolution strategies. IEEE CEC, 2008.
- [16] Y. Sun, D. Wierstra, T. Schaul and J. Schmidhuber, Stochastic search using the natural gradient. ICML, 2009.
- [17] Y. Akimoto, Y. Nagata, I. Ono and S. Kobayashi, "Bidirectional relation between CMA evolution strategies and natural evolution strategies", Proceedings of the 11th international conference on parallel problem solving from nature, 2010, Springer.
- [18] T. Glasmachers, T. Schaul, Y. Sun, D. Wierstra and J. Schmidhuber, "Exponential natural evolution strategies", GECCO 2010.
- [19] R. Storn, K. Price, "Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces". Journal of Global Optimization 11, 1997, pp. 341–359.
- [20] G. G. Yan, H. C. Chang, "Research on separable UEP-LT code", M.S. thesis, Dept. of Electron. Eng., National Chiao Tung University, Hsinchu, Taiwan, 2007, pp. 45–46.
- [21] M. E. H. Pedersen, "Good parameters for differential evolution", Technical Report no. HL1002, Hvass Laboratories, 2010.
- [22] R. Storn, "On the usage of differential evolution for function optimization". Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS), 1996.
- [23] O. Etesami, M. Molkaraie, A. Shokrollahi, "Raptor codes on symmetric channels", in Proceedings of International Symposium on Information Theory (ISIT), 2004, p. 38.
- [24] O. Etesami, A. Shokrollahi, "Raptor codes on binary memoryless symmetric channels," IEEE Transactions on Information Theory, vol. 52, no. 5, pp. 2033–2051, 2006.

Figure 4. Example of CMA-ES convergence in case that failure rate r is being optimized

Figure 5. Example of CMA-ES convergence in case that overhead ε is being optimized

Figure 6. Example of CMA-ES convergence in case that failure probability p is being optimized

In the experiments producing Figure 4 - Figure 12 we used population size equal to 20 and maximum number of evaluations equal to 10^4 .

Figure 7. Example of CMA-ES convergence with performance measurements setting: $p=0.0001$, $\epsilon=0.03$ and r being optimized

Figure 8. Example of NES convergence with performance measurements setting: $p=0.0001$, $\epsilon=0.03$ and r being optimized

Figure 9. Example of DE convergence with performance measurements setting: $p=0.0001$, $\epsilon=0.03$ and r being optimized

Note that the scale for Average degree is in DE case significantly larger than corresponding scale in CMA-ES and NES cases.

Figure 10. Example of CMA-ES convergence with initial degree distribution set to condensed robust soliton ($r=0$, $p=0.01$ and ϵ being optimized)

Figure 11. Example of CMA-ES convergence with initial degree distribution set to uniform ($r=0$, $p=0.01$ and ϵ being optimized)

Figure 12. Example of CMA-ES convergence with initial degree distribution set to sampled ideal soliton ($r=0$, $p=0.01$ and ϵ being optimized)