Multimed Tools Appl (2017) 76:15221–15250
DOI 10.1007/s11042-016-3756-9

Customizing short-length LT codes with evolution strategies for video streaming protection

Martin Hornansky[1] · John K. Zao[1]

Received: 1 July 2015 / Revised: 13 June 2016 / Accepted: 30 June 2016 /
Published online: 21 January 2017

Abstract Designing short-length Luby Transform (SLLT) codes to best protect video streaming and multicasting over lossy communication remains largely an empirical exercise. In this paper, we present a systematic approach to customize the decoding performance of these codes so that the protected video bitstreams may have the best playback quality over a wide range of channel loss rates. Our approach begins with the proposal of a new SLLT decoding performance model based on three parameters: decoding overhead, symbol decoding failure rate and tail probability of symbol decoding failure rate. We then formulate the design of SLLT codes as a multi-objective optimization problem, specify the design objectives in terms of goal program, and search for the most suitable codes using an augmented weighted Tchebycheff method implemented with the Covariance Matrix Adaptation Evolution Strategy (CMA-ES). Two design examples are provided to illustrate the effectiveness of our approach: (1) an SLLT post-code of a short-length raptor code that provides erasure protection to H.264 AVC bitstreams, and (2) an SLLT post-code of a rateless UEP code that supports graceful degradation of H.264 SVC playback quality. Empirical results demonstrate that the proposed method is capable of producing SLLT codes with customized decoding performance, whereas, the customized codes enable the playback pictures to attain significantly higher PSNR values at different stages of the decoding process than the pictures recovered under the protection of conventionally optimized codes.

Keywords Wireless multimedia streaming · Unequal erasure protection · LT codes · Raptor codes · Scalable video codes · Evolution strategies · Goal programming

✉ Martin Hornansky
hornansky.cs95g@nctu.edu.tw

John K. Zao
jkzao@cs.nctu.edu.tw

1 Department of Computer Science, National Chiao Tung University, 1001 University Road, Hsinchu, Taiwan 300, Republic of China

1 Introduction

Short-length rateless erasure correcting codes (a.k.a. *fountain codes*) with 10^2–10^4 symbols long code blocks are widely used in real-time data streaming and multicasting applications such as multi-point multimedia streaming [34], satellite communication [43], ad-hoc communication over wireless sensor networks [25] and free-space optical links [17]. The rateless decoding capability of these randomized codes enables them to recover data packets across a wide range of channel loss rates, whereas their short code blocks and linear decoding complexity make them suitable for protecting real-time low-power data transports. In most of these applications, however, the erasure protection offered by these rateless codes must be adapted to the rate-distortion characteristics of the source data and match the performance requirements of the application. Hyytiä et al. [13] suggested two basic performance objectives for designing short-length Luby transform (SLLT) codes: (1) minimizing the *average decoding overhead* for successful block decoding in the bulk data transport and (2) maximizing the *average symbol decoding success rate* at specific decoding overhead for the just-in-time decoding. Unfortunately, these objectives may not always reflect the demands of multimedia applications. For example, due to interdependency among groups of pictures (GOPs) in a compressed video bitstream, the fluctuation of rateless decoding failure rate among consecutive code blocks may deteriorate the video playback quality significantly even though the average symbol decoding success rate of the SLLT code was deemed acceptable [Section III.C]. Also, for the sake of protecting a scalable video code (SVC), the decoding performance of a *rateless unequal erasure protection* (*UEP*) *code* should have a tapered profile that is fine-tuned to match the rate-distortion performance of the SVC bitstream. Neither of Hyytiä's objectives can be used to produce such UEP codes.

Customizing an SLLT code to match a specific decoding performance profile is often a challenging task. This is because the relatively small number (10^2–10^4) of *input symbols* in the *code block* of an SLLT code may cause random high-order dependence relations to be formed among those symbols. The existence of these high-order dependence relations can deteriorate the decoding performance of the code far below the ideal behavior of its infinite-length variant and dashes any hope to develop an elegant theoretical performance model. In this paper, we present a general approach to customize the decoding performance of SLLT codes by adapting their sparse degree distributions using *evolution strategies*. Similar to the design of a digital filter [32], the design of customized SLLT codes can be formulated as a *multi-objective constrained optimization problem*. For the purpose of describing the non-Gaussian decoding performance of SLLT codes, we begin to tackle this task by proposing a new *statistical decoding performance model* based on three parameters: *decoding overhead*, *symbol decoding failure rate* and *tail probability of symbol decoding failure rate* [Section 3]. Then, we specify the performance requirements in terms of multiple *target points* in the *performance space* and adopt *weighted Tchebycheff goal programming* [14] to minimize the distance between the performance surface of the SLLT codes and specified set of target points [Section 4]. We chose to employ *Covariance Matrix Adaptation Evolution Strategy* (*CMA-ES*) [10] to search among the candidate codes and use Monte-Carlo simulation to estimate their decoding performance [Section 5].

Customizing SLLT codes often requires trading off their decoding performance in the transitional "waterfall" region and the steady "error floor" region. We demonstrate a way how to fine tune these tradeoffs through the design of an SLLT post-code of a short-length raptor code for protecting the H.264-AVC bitstreams [Section 6]. The design of SLLT codes for rendering unequal erasure protection to scalable video bitstreams posts even more demanding challenge: in order to implement graceful degradation of H.264-SVC playback quality over a wide-range of packet loss rates, we need to taper the decoding performance of the SLLT code so that it can match the rate-distortion profile of SVC bitstreams within a *region of interest*. Section 7 shows the design of an SLLT post-code that can be concatenated with a *convolutional UEP pre-code* to best provide the UEP service. The customized codes presented in Section 6–7 produce playback pictures with significantly higher PSNR values than those designed using the conventional optimization objectives.

2 Related work

The multimedia community has a long history in employing fountain codes: starting with the use of *systematic Raptor codes* in DVB-H standards for protecting IP datacasting [1], and soon followed by the applications in multimedia downloading [19] and scalable video multicasting [34]. The attempts to use short-length fountain codes in real-time scalable video streaming applications have also been made with the specific aim at optimizing their unequal erasure protection (UEP) profiles [3, 39]. However, because there is no known analytical method that can predict the decoding performance of fountain codes with practical lengths [39], researchers resorted to the use of numerical optimization techniques when searching for SLLT codes with desirable decoding behaviors. Shokrollahi and Storn were the first to use *stochastic optimization* to design finite-length fountain codes in 2000 [39]. Hyytiä et al. then tried to optimize the decoding performance of LT codes using *importance sampling* in 2006 [13]. More recently, Talari and Rahnavard used *multi-objective genetic algorithm* NSGA-II to design LT codes with good intermediate decoding performance [29]. We also tried to design composite UEP codes with SLLT post-codes using *evolutionary computation* techniques [33, 42].

When designing SLLT codes, most researchers chose to specify their target performance in terms of the *decoding overhead* and the *average decoding failure rate*. Several researchers, however, took a different approach by controlling the *expected output ripple size* throughout the decoding process: some tried to keep it constant [44]; others tried to increase its value [37] or reduce its variance [44]; yet others imposed a downward trend in order to suppress the occurrence of redundant connections late in the decoding process [40]. Unfortunately, due to the difficulty in correlating the decoding performance of an SLLT code with the trend of its expected ripple size, it is currently impractical to design SLLT codes by following this approach.

To make the search for candidate codes a little easier, some researchers tried to create short-length variants of the LT codes with asymptotically optimal *robust soliton distribution* (*RSD*). Attempts included tuning the RSD parameters δ and c [30], constraining its highest degrees [5] and adjusting the probability of those degrees that have the dominant influence on decoding performance [44]. However, all these attempts had only limited success due to the limits they impose on their degrees of freedom.

A key point seems to have been overlooked in most of these approaches, i.e. practical SLLT codes are usually required to match certain decoding performance profiles that are shaped by the rate-distortion characteristics of the source codes they are designed to protect. Thus, like a filter, an SLLT code must have its decoding performance *customized* with respect to the tradeoffs among different requirements rather than *optimized* on a certain aspect. These tradeoffs must be conducted in the form of multi-objective optimization within specified constraints. The LT decoding performance model proposed in the next section along with the goal programming approach and evolution strategy described henceforth provide the necessary framework for specifying the objectives and conducting the search for the best suited results.

3 LT decoding performance model

As mentioned, decoding overhead and average decoding failure rate are the two parameters commonly used to depict and specify the decoding performance of LT codes. Despite its simplicity, this model tends to mask the fact that decoding of LT code is a highly non-Gaussian random process. Hence, the average decoding failure rate and overhead may not be useful performance measurements of an LT code especially when its block length is small. As a prelude to the presentation of our multi-objective optimization strategy, we propose a tri-parametric performance model based on the tail probability of a random process. Furthermore, in order to reduce the dimensionality of the solution space, we propose to consider only the LT codes with sparse degree distributions that have compatible decoding performance with the dense codes.

3.1 Performance equivalent sparse degree distributions

Proposed by Michael Luby in 2002, the Luby Tnsform (LT) codes [20] were the first class of capacity-approaching rateless universal codes that fully realized the concept of *digital fountain* [24]. Using these codes, a sender can transmit unlimited number of *output symbols* until every receiver gets sufficient number of them and can thus decode the *input symbols* with high probability of success regardless of communication loss rates. Because their input symbols are uniformly randomly selected during the encoding process, the decoding performance of an LT code can be determined completely by the *degree distribution* of its output symbols, which can be expressed as a tuple of pairs $[(d,\ p_d)]_{d=1}^{d_{\max}}$ with d being the *degree* of an output symbol and p_d its *probability* of occurrence.

There is sufficient evidence that LT codes, which have only few degrees with non-zero probability, can exhibit decoding performance similar to that of their dense counterparts [13]. These LT codes with *sparse degree distributions* can be specified in terms of two tuples:

1. *Degree Tuple*: $\mathbf{d} \underline{\underline{\mathrm{def}}} [d_i]_{i=0}^{M-1}$ with $d_i \in [1, K]$. and $d_0 = 1$
2. *Probability Tuple*, $\mathbf{p} \underline{\underline{\mathrm{def}}} [p_i]_{i=0}^{M-1}$ with $p^i \in [0, 1]$ $\sum_{i=0}^{M-1} p^i = 1$

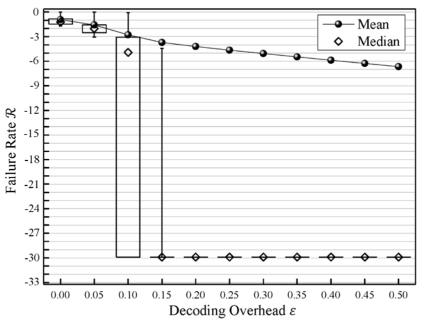
where K is the *block length*, i.e., the number of input symbols in the code; and M is the *cardinality* of the code's degree distribution, i.e., the maximum number of degrees with non-zero probability. Elements in these two tuples are the *design variables* that we used to customize the decoding performance of the SLLT codes.

An important advantage of employing SLLT codes with sparse degree distributions is that the customization of their decoding performance can be carried out within feasible time span using evolutionary computation techniques, which can handle non-convex and ill-conditioned optimization problems [38]. Chen et al. developed a method that can produce a sparse variant of any dense LT code with practically identical decoding performance [8]. We used this tool and RSD to determine the initial values of the degree and probability tuples.

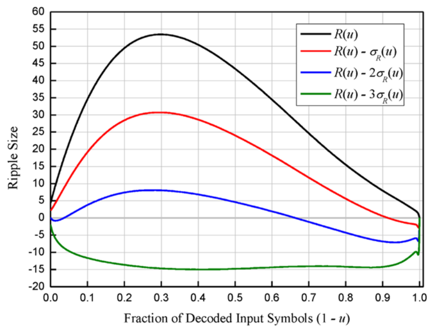
3.2 Multi-modal probability distribution of decoding performance

Due to the random selection of input–output symbol connections during the encoding process, each LT code can be represented as an overdetermined system of linear equations specified by a random binary connectivity matrix. As shown in Fig. 1a, such a system may yield dispersed decoding failure rates (especially in the cases of short block length) when the decoding process is carried out with small decoding overhead ε and the system is just slightly overdetermined. This performance discrepancy can be explained by the large variation of expected output ripple size throughout the decoding process (Fig. 1b), which makes the probability that output ripple size becomes zero (i.e., probability that decoding process terminates) in the early stage of the decoding process rather high. The occurrence of the failure cases dramatically increases in SLLT codes as the shortening of their code blocks may result in forming many more random high-order dependence relations among their input symbols, which often hamper successful decoding process. Note that in Fig. 1b we estimated the expected output ripple size using the technique proposed by Shokrollahi et al. [21].

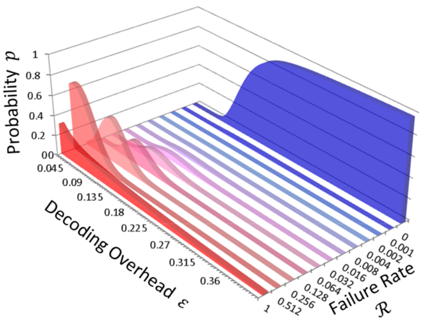
Based on the above description, we may treat the decoding failure rate r of the input symbols within an SLLT code block as a random variable parameterized with respect to its decoding overhead ε. Figure 1c shows the logarithmic profile of the *discrete failure probability*



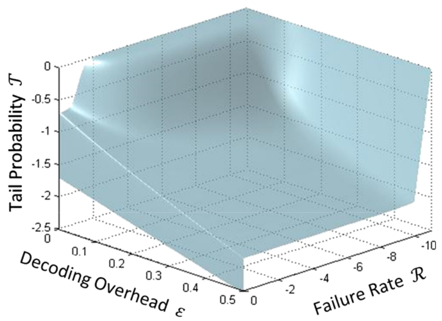
(a) Boxplot of *symbol decoding failure rate* $\mathcal{R}$ showing mean, median, IQR and the 10%, 90% whiskers



(b) Estimated variation of *expected output ripple size* during the decoding process



(a) Log-profile of *discrete probability distribution* $p_\varepsilon(r)$ of *decoding failure rate* r w.r.t. overhead ε



(b) Log-log profile of *tail probability* $\mathcal{T}$ of *decoding failure rate* $\mathcal{R}$ w.r.t. overhead ε

Fig. 1 Decoding performance of SLLT code that adopts *robust soliton distribution* with $K = 1000$, $\delta = 0.5$ and $c = 0.0125$

distribution $p_\varepsilon(r)$ of one SLLT code while Fig. 1d shows the log-log profile of the corresponding *tail probability* $T_\varepsilon(r) \stackrel{\text{def}}{=} 1-P_\varepsilon(r)$ with $P_\varepsilon(r)$ being the *cumulative failure probability distribution*. The bi-modal nature of the failure probability distribution (especially in the low overhead region) is obvious.

3.3 Decoding performance parameters

Since the first and second order moments often fail to accurately depict the nature of non-Gaussian probability distributions, we adopted the approach of statistical hypothesis tests, i.e. we based our performance model on the *tail probability* of the target random variable—in this case, the *symbol decoding failure rate*. The benefit of employing the tail probability instead of using mean and variance is that it takes into account the probability of *outliers* without being affected by their extreme values. Hence, we based our SLLT code performance model on the following three parameters.

Definition 1: *Decoding Overhead* ε: it is the ratio between the number of extra output symbols received and the number of input symbols in the code block.

Definition 2: *Symbol Decoding Failure Rate* $r_{\hat{\varepsilon}}$ *at specific decoding overhead* $\hat{\varepsilon}$: it is the fraction of the input symbols within the code block that failed to be decoded after $K{\cdot}(1+\hat{\varepsilon})$ output symbols have been used in the decoding process; this parameter shall be treated as a random variable parameterized with respect to the decoding overhead.

Definition 3: *Tail Probability* $T_{\hat{\varepsilon}}(\hat{r})$ *of a specific decoding failure rate* $\hat{r}$ *at a specific decoding overhead* $\hat{\varepsilon}$: it is the probability that the decoding failure rate exceeds a specific value $\hat{r}$ when the output symbols are decoded with overhead $\hat{\varepsilon}$.

In order to exaggerate the targeted value ranges of these parameters, we chose to adopt the common practice of expressing some of the parameters in logarithmic scale. Specifically, we employed $\mathcal{R}_{\hat{\varepsilon}} \stackrel{\text{def}}{=} \log_2 r_{\hat{\varepsilon}}$, the base-2 logarithm of the symbol failure rate and $\mathcal{T}_{\hat{\varepsilon}}(\hat{\mathcal{R}}) \stackrel{\text{def}}{=} \log_{10}\left(1-P_{\hat{\varepsilon}}\left(\hat{\mathcal{R}}\right)\right)$, base-10 logarithm of its tail probability.

4 SLLT code customization approach

The decoding performance parameters (ε, $\mathcal{R}$, $\mathcal{T}$) defined in Section 3 allow a code designer to specify multiple performance objectives that can be imposed on the SLLT code in order customize its decoding behavior. However, unlike in the case of single-objective optimization, multi-objective optimization often yields a set instead of a single optimal solution. The set of all optimal solutions is called *Pareto front* and its characteristic property is that none of its elements can be improved with respect to any objective without being degraded with respect to at least one another [21], and therefore, all of its members must be considered equally good. In order to find the most suitable member of the Pareto front, the user must specify a quantified preference (or relative importance) among multiple objectives and employ a *scalarization* technique to combine them into a single objective according to that quantification. In this section, we present the rationale of employing *distance-based goal programming* [14] as the scalarization technique for SLLT code customization and provide a template for specifying the decoding performance targets.

4.1 Distance based goal programming

The *distance based goal programming* is intuitive, relatively easy to implement and very commonly employed technique that converts the multi-objective optimization problem into the single-objective one. In the distance based goal programming, multiple objectives/goals are specified as one goal point and the optimization problem is formulated as minimization of an achievement function that measures the distance to this goal point. The specific way, in which this distance is measured, defines the specific variant of the goal programming, whereas, the *achievement function* $F(\mathbf{x})$ defined in a *search space* $\boldsymbol{X}$ has the following general form:

$$F(\mathbf{x}) \triangleq \|\boldsymbol{F}(\mathbf{x})-\boldsymbol{G}\|_{\mathcal{p}}^{\boldsymbol{w}} \overset{\text{def}}{=} \left\{\sum_{\ell=1}^{\mathcal{L}}\left(w_{\ell}\cdot|F_{\ell}(\mathbf{x})-G_{\ell}|\right)^{\mathcal{p}}\right\}^{1/\mathcal{p}} \tag{1}$$

In above expression, $\mathbf{x}$ represents a candidate solution, which in our case corresponds to a generated SLLT code (uniquely specified through its degree distribution in the space of all admissible SLLT codes), $F_{\ell}(\mathbf{x})$ is an *achieved value* of the solution $\mathbf{x}$ on the ℓ-th objective, G_{ℓ} denotes numeric *target level* of the ℓ-th objective, w_{ℓ} is its positive weight, $\mathcal{p}$ represents the order of employed norm and $\mathcal{L}$ denotes the number of objectives/goals.

When adopting the proposed statistical performance model, a single objective imposed on the decoding performance of an SLLT code can be seen as one target point $P_j = \left(\hat{\varepsilon}_j, \hat{\mathcal{R}}_j, \hat{\mathcal{T}}_j\right)$ in the decoding performance space that should be attained as closely as possible along specific axis, and therefore, the target and achieved value of the ℓ-th objective can be specified as follows:

- $G_{\ell} \in \left\{\hat{\varepsilon}_j, \hat{\mathcal{R}}_j, \hat{\mathcal{T}}_j\right\}$ representing value of chosen coordinate of the corresponding target point P_j determined by the axis, along which the distance between this target point and decoding performance surface of SLLT code should be minimized according to the ℓ-th performance objective;
- $F_{\ell} \in \left\{\varepsilon_j, \mathcal{R}_j, \mathcal{T}_j\right\}$ representing value of chosen coordinate of the point P_j' located on the decoding performance surface of candidate SLLT code, whereas, the other two coordinates of this point are identical with coordinates of the point P_j, representing the corresponding target point. In other words, the point P_j' can be defined as follows:

  - If $G_{\ell} = \hat{\varepsilon}_j$: $P_j' = \left(\varepsilon_j, \hat{\mathcal{R}}_j, \hat{\mathcal{T}}_j\right)$;
  - If $G_{\ell} = \hat{\mathcal{R}}_j$: $P_j' = \left(\hat{\varepsilon}_j, \mathcal{R}_j, \hat{\mathcal{T}}_j\right)$;
  - If $G_{\ell} = \hat{\mathcal{T}}_j$: $P_j' = \left(\hat{\varepsilon}_j, \hat{\mathcal{R}}_j, \mathcal{T}_j\right)$.

Two norms are commonly used to measure the distance between the goal point and achieved fitness value vector of a candidate solution: *Manhattan norm* ($\mathcal{p} = 1$) and *infinite norm* ($\mathcal{p} = \infty$); which are used in *Archimedean* [14] and *Tchebycheff goal programming* [24], respectively.

1) Tchebycheff Goal Programming

One very beneficial property of goal programming variants with finite norms is that they allow to directly trade-off different objectives by adjusting the corresponding weights, but on the other hand, the Tchebycheff goal programming, which aims at balancing the non-achievement of all the goals, is the only distance-based goal programming variant that possesses a capability to generate solutions located in the non-convex parts of the Pareto fronts [16]. Since according to our empirical results the Pareto fronts of optimized SLLT codes often exhibit non-convex regions [12], we suggest to adopt the Tchebycheff goal programming when customizing their decoding performance. Note that since the Tchebycheff goal programming adopts the infinite norm, the general form of its achievement function can be reformulated as follows:

$$F(\mathbf{x}) \triangleq ||\boldsymbol{F}(\mathbf{x}) - \boldsymbol{G}||_{\infty}^{\boldsymbol{w}} = \max_{1 \le \ell \le \mathcal{L}} \{ w_\ell \cdot |F_\ell(\mathbf{x}) - G_\ell| \} \tag{2}$$

It can be seen that expressions (1) and (2) penalize both underachievement as well as overachievement of each goal. In many cases, however, penalizing the overachievement may not be desirable, especially when solutions satisfying all the goals may exist [8]. The Tchebycheff goal programming can easily reward the case of overachievement, in which all the goals are satisfied, by replacing the absolute value in (2) with round brackets.

2) Augmented Weighted Tchebycheff Method

By definition, the Tchebycheff goal programming yields always a solution that cannot be improved w.r.t. the objective that exhibits maximum difference between the achieved value $F_\ell(\mathbf{x})$ and target value G_ℓ without being degraded w.r.t. some other objective(s). However, since this difference does not contain precise information about the corresponding differences on other objectives, all the solutions with the same value of the maximum difference are considered equally good in this goal programming variant. As result, the Tchebycheff goal programming may sometimes yield *weakly optimal Pareto solution*, i.e., a solution that cannot be improved w.r.t. the objective that exhibits maximum difference between its achieved and target value but can be improved with respect to some other objective(s). In order to avoid this shortcoming, we employed in examples described in Sections 6 and 7 *augmented weighted Tchebycheff method* [12] that was specially designed to eliminate this undesirable property by augmenting the infinite norm with L_1-term multiplied by "small" parameter ϵ, called *augmentation coefficient*:

$$\min_{\mathbf{x} \in X} \mathbf{F}(\mathbf{x}) \triangleq \max_{1 \le \ell \le \mathcal{L}} \{ w_\ell \cdot [F_\ell(\mathbf{x}) - G_\ell] \} + \epsilon \cdot \sum_{\ell=1}^{\mathcal{L}} [F_\ell(\mathbf{x}) - G_\ell] \tag{3}$$

Specification 1 Goal Program Template

```
$J$ in $\mathbb{N}$                                  // Number of basic performance blocks (i.e., target points)
for $j = 1$ to $J$ define
  Performance_Block$_j$ {
    $Target = (\hat{\varepsilon}_j, \hat{\mathcal{R}}_j, \hat{\mathcal{T}}_j)$       // Coordinates of target point $P_j$
    $Goals$ in {1,2,3}                          // Number of goals associated with $P_j$
    for $h = 1$ to $Goals$ define
      $(g_h, w_h)$ in $(\{\varepsilon, \mathcal{R}, \mathcal{T}\}, [0,1])$   // Tuple of goal and corresponding weight
  }
$Norm\_Degree$ in $\{1, \infty\}$                    // Distance norm of adopted goal programming variant
}
```

4.2 Target performance specification

In order to design proper goal program when customizing the SLLT code performance using the statistical performance model defined in Section 3, a code designer must specify the desired decoding performance in terms of J target points $P_j \triangleq \left(\hat{\varepsilon}_j, \hat{\mathcal{R}}_j, \hat{\mathcal{T}}_j\right)$, whereas, each of these target points must have assigned one, two or three goals representing axes, w.r.t. which the distance between the corresponding target point and decoding performance surface of the SLLT code should be minimized (Fig. 2). Moreover, the code designer must assign to each goal G_ℓ a weight w_ℓ representing its priority and choose the norm, according to which the weighted distance to the goal point $\boldsymbol{G}$ shall be measured [Specification 1].

The proposed method may be perceived as a process that pulls the performance surface of the initial SLLT code candidate towards the set of target points, whereas, improved performance in certain region(s) of the decoding performance space is usually achieved at the cost of weaker performance in some other region(s). Note that the proposed method provides a code designer the choice to specify the performance objectives also in terms of the *average failure rate* and *average overhead*, however, the important benefit of specifying the performance objectives in terms of the target points in the $(\varepsilon, \mathcal{R}, \mathcal{T})$-space is that it enables the employed optimization method to coerce the SLLT code to approximate any performance profile.

5 Adaptation of covariance matrix adaptation – evolution strategy (CMA-ES)

CMA-ES [10] is a popular *non-elitist evolution strategy*, which was particularly designed to solve *non-convex* and *ill-conditioned* continuous optimization problems possessing noisy fitness function(s) with unknown geometry. Hansen and Kern showed that CMA-ES exhibits more robust and efficient convergence behavior than three widely used stochastic optimization methods: *differential evolution* (DE), *robust evolution strategies* (RES) and *local optima smoothing* (LOS), on a set of classical multimodal and *non-*

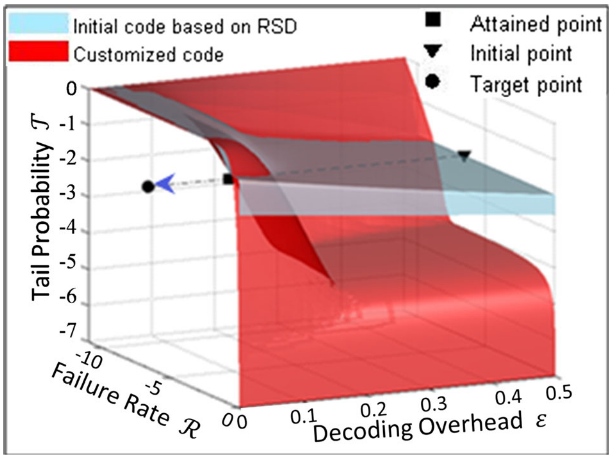


Fig. 2 Decoding performance comparison of initial solution and customized SLLT code aiming at attaining the target point $P = [0.13, -6.64, -2]$

separable test functions [4]. Furthermore, it has been proven that the updates of the mean and covariance matrix of the sample distribution in the CMA-ES follow the maximal log-likelihood approach, while its iterative search resembles the *natural gradient descent* over expected objective function value [23]. All these facts provide further support to the claim of its robust and efficient convergence behavior. Note that the efficient convergence behavior is of crucial importance for the SLLT code customization problem because the evaluation of the decoding performance is carried out using computationally expensive *Monte Carlo* simulations.

5.1 Population size and initial solution

When employing the CMA-ES, a user has to set the values of two parameters: *population size* and *initial solution*. In the proposed method, we adopted the recommended population size $\lambda = 4 + \lfloor 3 \cdot \ln(n) \rfloor$ suggested in [9], where n represents the number of decision variables, in our case corresponding to the maximum number of degrees with non-zero probabilities multiplied by 2. The recommended population size of CMA-ES is relatively small comparing to the recommended population size of other methods belonging to the class of *evolutionary computation*. Moderate population size is very appreciated when solving optimization problems with computationally expensive fitness function evaluations because it usually significantly affects the convergence speed.

Concerning the choice of the initial solution, we based our design on a sparse variant of the *robust soliton distribution* (RSD) [20] because of its wide use in the design of finite-length LT codes [18]. The robust soliton distribution $\mu(d)$ has an adjustment term $\tau(d)$ added to the asymptotically optimal *ideal soliton distribution* $\rho(d)$ in order to increase its expected ripple size η, i.e., a set of output symbols of reduced *degree-1* at the specific decoding step [7]. In addition, RSD possesses two parameters influencing its slope and position of the second peak, called *spike*. One of these parameter is a *free parameter c* that represents tradeoff between the overhead efficiency and decoding delay [30], while the other one, denoted δ, represents the upper bound of the probability for the ripple to vanish if the number of received symbols approaches $K + \mathcal{O}\left(\sqrt{K} \cdot \ln^2(K/\delta)\right)$ [20].

$$\mu(d) \triangleq \frac{\rho(d) + \tau(d)}{\sum_{d=1}^{K} (\rho(d) + \tau(d))} \tag{4}$$

$$\rho(d) \triangleq \begin{cases} 1/K & d = 1 \\ 1/(d \cdot (d-1)) & d = 2, \ldots, K \end{cases} \tag{5}$$

$$\tau(d) \triangleq \begin{cases} \eta/(K \cdot d) & d = 1, \ldots, \lfloor K/\eta - 1 \rfloor \\ (\eta/K) \cdot \ln(\eta/\delta) & d = \lfloor K/\eta \rfloor \\ 0 & d > \lfloor K/\eta \rfloor \end{cases} \tag{6}$$

In order to create a proper initial sparse degree distribution, we adopted an approach where we first chose reasonable values of the RSD parameters δ and c, and then deduced the sparse (d_i, p_i) value pairs. Since in several SLLT code designs for multimedia and real-time applications [30] authors employed the RSD with $\delta = 0.5$, we adopted this setting in our design of the initial solution, too. On the other hand, when considering the parameter c, it is important to note that this parameter has a strong influence on the expected ripple size η, since:

$$\eta = c \cdot \left(\ln(K/\delta) \cdot \sqrt{K} \right) \tag{7}$$

The successful decoding process requires $\eta \geq 1$, however, too large η causes a waste of the input symbols by increasing the redundancy of their coverage. Although, the decoding process is more likely to prematurely terminate when η is small [11], the SLLT codes achieve the best performance when c is close to its lower bound [30]. Taking into consideration these facts, we adopted $\eta = 4$, when setting the value of the parameter c.

In order to choose the elements of the degree tuple $\mathbf{d}$, we used the *tags selection function* [38] and considered only the range between the *degree-1* and spike $\hat{d}$ of adopted RSD, since in [5] authors showed that the degrees beyond the spike can be ignored without harming the performance of the LT code. Consequently, in order to make each selected tag to be a proper representative of its range, we set the corresponding p_i value to the cumulative probability of all the degrees belonging to its range, while as range boundaries we set the logarithmic midpoints between two adjacent tags [33].

5.2 SLLT code design variables vs. CMA-ES object variables

Based on the fact that the solution space of admissible sparse degree distributions is highly constrained, while the search space of the CMA-ES object variables is continuous and unbounded, we defined two tuples of search space variables: (1) tuple of elements corresponding to the permissible degrees: $\mathbf{x} \underline{\underline{\mathrm{def}}} \ [x_i]_{i=0}^{M-1} \in \mathbb{R}^M$; and (2) tuple of elements corresponding to their respective probabilities $\mathbf{y} \underline{\underline{\mathrm{def}}} \ [y_i]_{i=0}^{M-1} \in \mathbb{R}^M$.

In order to ensure that the permissible degrees $d_0, \ldots, d_{M-1}$ of each SLLT code have always valid values, we map each element of the $\mathbf{y}$ tuple into a discrete value from the admissible interval $[1, K]$ whenever we sample a new candidate solution.

$$d_i^{(g)} = \begin{cases} 1 & x_i^{(g)} < 1 \\ \left\lfloor x_i^{(g)} + 0.5 \right\rfloor & 1 \leq x_i^{(g)} \leq K \\ K & x_i^{(g)} > K \end{cases} \tag{8}$$

In above expression, g denotes the current generation number.

On the other hand, it is important to note that the probability tuple $\mathbf{p}$ exhibits only $M-1$ degrees of freedom since $\sum_{i=0}^{M-1} p_i = 1$, whereas, the valid values of its elements belong to the interval [0,1]. In order to comply with these facts as well as with the *Gaussian distribution* of the unbounded search space, we employ following feasibility-preserving quadratic transformation when sampling a new candidate code:

$$p_i^{(g)} = \frac{\left(y_i^{(g)}\right)^2}{\sum_{i=0}^{M-1} \left(y_i^{(g)}\right)^2}, \qquad i = 0, \ldots, M-1 \tag{9}$$

5.3 Design of fitness penalties and stopping criteria

In the proposed approach, the distance of each generated candidate solution from the feasible space of valid degree distributions is quantified using two monotonically increasing quadratic penalty functions [27], whose sum is added to its fitness value (Table 1). The first penalty function penalizes the candidate solutions with elements of the tuple **x** located outside of the range of *feasible degrees*, whereas, the second penalty function penalizes the candidate solutions depending on the distance of their probability tuple elements from the lower boundary of the *feasible probability* space, i.e., $1/K$. We implemented this penalty function based on the fact that the number of output symbols used to decode the code block is usually very close to K, which means that the receiver is likely to receive only output symbols that have degrees with the probability at least $1/K$.

In Table 1, $]x-y]$ and $[x-y[$ represent the *left-side difference* and *right-side difference*, respectively.

$$
\begin{aligned}
]x-y] &= \begin{cases} |x-y| & \text{for} \quad x \geq y \\ 0 & \text{for} \quad x < y \end{cases} \\
[x-y[&= \begin{cases} 0 & \text{for} \quad x \geq y \\ |x-y| & \text{for} \quad x < y \end{cases}
\end{aligned}
\tag{10}
$$

When employing a *stochastic optimization method*, it is necessary to design appropriate stopping criteria in order to avoid premature termination of the search process as well as wasting the computational resources. Since the formulated SLLT code customization problem possesses computationally expensive fitness function evaluations, the importance of proper choice of the termination rules becomes even more crucial. In the proposed approach, we adopt a stopping criterion that represents the conjunction of the fitness value and design variable convergence because this kind of *combined stopping criteria* has been shown to be advantageous when employed with the *evolutionary optimization methods* [6]:

$$
\left| f_{i,min}^{(g-\Delta g)} - f_{i,min}^{(g)} \right| < \frac{f_{i,min}^{(g)}}{\varphi} \ \forall i \quad \wedge \quad \sum_{j=0}^{M-1} \frac{\sqrt{\mathrm{Var}[d_j \cdot p_j]}}{E[d_j] \cdot E[p_j]} < \frac{\mathrm{E}[D]}{\omega}
\tag{11}
$$

In the above expression, $f_{i,min}^{(g)}$ denotes the minimum value of the fitness function i in the generation g, Δg represents the time window during which the fitness value improvement is observed, $\mathrm{E}[D]$ is the expected (i.e., average) degree, and φ with ω are the parameters determining the thresholds of the fitness value and design variable convergence. In order to prevent unfeasibly long evolution loops in cases when above defined combined stopping

Table 1 Employed fitness penalty functions

Term	Formulae
Quadratic penalty function for out-of-range degrees	$G(\mathbf{x}) = \sum_{i=0}^{M-1}]x_i - K]^2 + \sum_{i=0}^{M-1} [x_i - 1[^2$
Quadratic penalty function for negligible probabilities	$G(\mathbf{p}) = \sum_{i=0}^{M-1} \left[\frac{\log p_i - \log \frac{1}{K}}{\log \frac{1}{K}} \right[^2$
Transformed fitness function	$\tilde{F}(\mathbf{d}, \mathbf{p}) = F(\mathbf{d}, \mathbf{p}) + (C \cdot g)^{\alpha} \cdot [G(\mathbf{p}) + G(\mathbf{x})]$

criterion fails to be satisfied, we implemented an additional stopping condition ensuring that the optimization process terminates no later than after a specific sufficiently large constant number of candidate solution evaluations:

$$N_{eval} > \varrho \tag{12}$$

In above expression, N_{eval} and ϱ denote the actual and maximum number of candidate solution evaluations.

5.4 Recommended optimization settings

Since our empirical results showed us that practically every dense degree distribution of SLLT code has an equivalent sparse counterpart with no more than 7 non-zero degrees, we believe that it is appropriate to set the cardinality of the sparse degree distributions $M = 7$. In addition, it is reasonable to keep $d_0 = 1$ and $d_1 = 2$ throughout the entire optimization process because the non-zero probabilities of *degree-1* and *degree-2* are essential for the successful decoding [15, 44]. The number of encoding/decoding simulation runs used to evaluate the performance of generated candidate SLLT code should be set based on the length of the code block and desired estimation precision, whereas, if the code block length $K \sim 1000$, we suggest to evaluate each candidate solution using 10^4 simulation runs. In Table 2, an interested reader can see the penalty function parameters and augmentation coefficient value as suggested in [45], along with the recommended stopping condition thresholds, which are based on our empirical experience.

6 Customizing transitional and steady decoding performance of SLLT codes

6.1 Customization strategies in transitional and steady decoding performance regions

Generally, the decoding behavior of an SLLT code can be characterized in terms of its performance in the *transitional* and *steady* regions of the decoding process. Based on this fact, the customization of an SLLT code may be conducted through a constrained optimization using a set of *goals* specified in these two decoding performance regions in terms of performance parameters $(\varepsilon, \mathcal{R}, \mathcal{T})$. However, it is important to note that a goal specified in one region may not have only very different effect on the overall decoding performance comparing to a goal specified in another region but specified goals may often also compete with one another. Based on these facts, a code designer must clearly understand the effects of goal setting and assign each of them a weight reflecting its relative priority or importance.

Table 2 Recommended numerical simulation settings

Penalty function parameters		Stopping condition parameters				Augmentation coefficient
α	C	Δg	φ	ω	ϱ	ϵ
2	0.5	20	10^2	10^3	10^4	10^{-2}

6.1.1 Transitional decoding performance region

In this region, the symbol decoding failure rate $\mathcal{R}$ begins to drop gradually starting at a relatively small decoding overhead ε. Nonetheless, the decoding failure rate in this region fluctuates drastically among decoding runs (Fig. 1a) resulting in a bi-modal probability distribution. Based on these facts, the performance objectives in the transitional region of decoding performance usually take one or both of the following two forms, while maintaining a tolerable level of the tail probability $\hat{\mathcal{T}}$:

a) Minimize the positive distance between the actual and target value of *decoding overhead* $]\varepsilon{-}\hat{\varepsilon}]$ that yields a tolerable symbol failure rate $\hat{\mathcal{R}}$ representing start or center of the transitional region;
b) Minimize the positive distance between the actual and target value of *symbol decoding failure rate* $]\mathcal{R}{-}\hat{\mathcal{R}}]$ at a specific low level of decoding overhead $\hat{\varepsilon}$, whereas $\hat{\mathcal{R}}$ represents the start or center of the transitional region.

Please note that the symbol decoding failure rate r of a single run changes in discrete steps of $1/K$ with K being the block size and reducing it at a low overhead ε may cause it to rise at a higher overhead, i.e. in the steady decoding performance region.

6.1.2 Steady decoding performance region

In this region, the symbol decoding failure rate r asymptotically approaches its lower bound as the decoding overhead ε continues to rise. During this process, the fluctuation of failure rate between different decoding runs also diminishes, which causes the tail probability $\mathcal{T}$ to fall. Based on these facts, the performance objectives in the steady region of decoding performance may take one or more of the following forms:

a) Minimize the positive distance between the actual and target value of *decoding overhead* $]\varepsilon{-}\hat{\varepsilon}]$ yielding an expected floor value $\hat{\mathcal{R}}$ of the symbol failure rate with certain tolerable tail probability $\hat{\mathcal{T}}$;
b) Minimize the positive distance between the actual and target value of *symbol decoding failure rate* $]\mathcal{R}{-}\hat{\mathcal{R}}]$ yielding certain tolerable tail probability $\hat{\mathcal{T}}$ at overhead $\hat{\varepsilon}$ that marks the beginning of the steady decoding performance region;
c) Minimize the positive distance between the actual and target value of *tail probability* $]\mathcal{T}{-}\hat{\mathcal{T}}]$ at a relatively high overhead $\hat{\varepsilon}$ and a zero-equivalent[1] decoding failure rate $\hat{\mathcal{R}}\triangleq\log_2[1/(K{\cdot}2)]$.
d) Minimize the positive distance between the actual and target value of *average symbol failure rate* $]\overline{\mathcal{R}}{-}\hat{\mathcal{R}}]$ at certain moderate level of decoding overhead $\hat{\varepsilon}$.

Note that since the *average* failure rate $\overline{\mathcal{R}}$ is the expected value of symbol failure rate that can be among others derived also from the tail probability of individual failure rates $\hat{\mathcal{R}}$= {0, 1/

[1] Since zero decoding failure rate equals to $-\infty$ in the logarithmic scale, we set an artificial lower bound of $\mathcal{R}$ based on the length of the code block K

K, $2/K$, …, 1}, the last goal is effectively a bi-variate objective based on the common concept of average failure rate.

6.2 Procedure for developing goal programs

In multimedia applications, the SLLT codes are customized to protect source codes with specific rate-distortion performance profiles and to meet certain performance objectives. Thus, it is essential to possess a systematic way to devise the goal programs capable to customize the SLLT code performance based on the source code characteristics and design objectives. The following four-step procedure offers such a way.

a) Identify the performance profiles of the source code and/or pre-code:

   - Determine rate-distortion performance of the source code, for example, by simulating the distortion ratios of different NAL units in SVC-MGS groups of pictures on various multimedia benchmarks;
   - Estimate erasure capabilities of the pre-code, for example, by simulating its decoding behavior in *random erasure channel*.

b) Formulate performance objectives of the SLLT code: Specify desired performance characteristics of the SLLT code in terms of the optimization objectives defined in the $(\varepsilon, \mathcal{R}, \mathcal{T})$-space.
c) Translate performance objectives into goal program blocks: Map each formulated performance objective into a target point and optimization goal representing the axis, along which the distance to the target point should be minimized.
d) Assign weights to individual goals according to their priorities: setup a weight vector representing the importance/priority of each goal with respect to the desired performance behavior.

In this section, we use a design of the SLLT post-code of short-length raptor code as an example to demonstrate the way of carrying out this procedure.

6.3 Raptor codes

As high-performance realization of *fountain codes*, the raptor codes [28, 41] concatenate an LT post-code with high-rate pre-code. The advantage of *pre-coding* is that instead of having to recover all the input symbols with the LT decoder, only a large fraction of them needs to be recovered, since the built-in redundancy of the pre-code can be used to recover the remaining ones [31]. Shokrollahi in his seminal paper [7] offered a heuristic to generate an LT post-code of the raptor code, but regrettably, his heuristic fails to take into account the erasure recovery capabilities of adopted pre-code. To address this shortcoming, in subsequent sections we describe a search for customized SLLT post-code that matches erasure protection capabilities of chosen pre-code employing proposed approach along with specified procedure for developing goal programs. The goal of short-length raptor that concatenates customized SLLT post-code with chosen high-rate pre-code is to provide GOP-based erasure protection to low-rate H.264 AVC bitstreams with minimum decoding overhead and low GOP decoding failure rate.

6.4 Design of SLLT post-code for raptor code protecting AVC streaming

Specification 2 Goal Program Setup to Protect AVC Streaming

```
$\mathcal{J} = 2$
Performance_Block$_1$ {
    $Target = (0, -4.32, -2)$          // $\hat{r} = 0.05$ and $\hat{T} = 0.01$
    $Goals = 1$
    $(g_1, w_1) = (\varepsilon, 0.5)$
}
Performance_Block$_2$ {
    $Target = (0.1, -4.32, -5)$        // $\hat{r} = 0.05$ and $\hat{T} = 0$
    $Goals = 1$
    $(g_1, w_1) = (\mathcal{T}, 0.5)$
}
$Norm\_Degree = \infty$
```

6.4.1 Choice and performance of adopted pre-code

In order to be able to recover most of the symbols after receiving only small decoding overhead, the pre-code of the raptor code should possess the ability to provide high rate when the code block length K is short. Based on this fact, we decided to employ the *low-density parity-check* (*LDPC*) *code* [22] as the pre-code of our raptor code. More precisely, we employed the 1998.5.3.1392 LDPC code with the rate equal to 0.889 when the code block contains 1998 symbols, provided by MacKay in his database of sparse graph codes [2]. Figure 3 shows the decoding performance of adopted pre-code in a statistically independent *random erasure channel*.

6.4.2 Performance objectives

In the multimedia streaming, a code designer usually wants to achieve decent video quality after receiving the smallest possible number of symbols and full decoding quality at a specific moderate level of decoding overhead. Based on these facts, we imposed on the decoding performance of designed SLLT post-code following performance objectives:

- Minimize the decoding overhead ε yielding maximum tolerable failure rate $\hat{\mathcal{R}}_{max}$ with maximum tolerable tail probability $\hat{\mathcal{T}}_{max}$ guaranteeing decent video quality;
- Minimize the tail probability $\mathcal{T}$ of maximum tolerable failure rate $\hat{\mathcal{R}}_{max}$ at a specific moderate (i.e., maximum acceptable) level of decoding overhead $\hat{\varepsilon}_{max}$.

Note that minimization of the decoding overhead and tail probability corresponds to the minimization of the positive distance between the actual value of the decoding overhead and its target value $\hat{\varepsilon}_{zero} = 0$, i.e. between the actual value of the tail

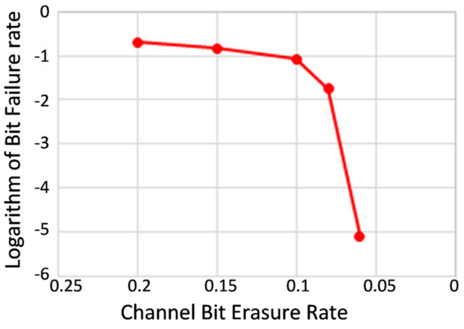


Fig. 3 Decoding performance of adopted LDPC pre-code in random bitstream

probability and its target value $\hat{\mathcal{T}}_{zero} = \log[1/(N \cdot 10)] = -5$, representing its artificial lower bound. [2]

6.4.3 Target points and optimization goals

In order to specify the coordinates of target points representing the defined performance objectives, it is necessary to find out the values of following thresholds:

- Maximum tolerable failure rate $\hat{\mathcal{R}}_{max}$ determining the erasure rate, at which is adopted pre-code capable to recover practically all the source data;
- Maximum tolerable tail probability $\hat{\mathcal{T}}_{max}$ (of maximum tolerable failure rate) representing the maximum frequency of code blocks that may fail to be recovered while delivering acceptable video quality;
- Decoding overhead $\hat{\varepsilon}_{max}$, at which is the SLLT post-code expected to deliver decoding quality enabling the raptor code to provide practically full video quality.

One of the ways how to determine the failure rate range, at which is adopted LDPC pre-code capable to recover practically all the source data, is to simulate its symbol recovery process in a random erasure channel. Since our experiments show that adopted LDPC pre-code is capable to recover practically all the erased symbols as long as the *channel erasure rate* does not exceed 5 % (Fig. 3), we set the threshold of tolerable failure rate $\hat{\mathcal{R}}_{max}$ to 0.05.

On the other hand, the maximum frequency of code blocks that may fail to be recovered while allowing the raptor code to deliver decent video quality, depends mostly on the characteristics of protected data. Since according to our experience, the quality of the video playback can be considered acceptable as long as there is no more than 1 out of 100 blocks that fails to be fully recovered, we set $\hat{\mathcal{T}}_{max} = 0.01$ as the maximum tail probability of tolerable failure rate $\hat{\mathcal{R}}_{max}$.

When determining the value of the decoding overhead, at which is expected the full video quality, one should take into consideration the length of the code block K and real-time requirements of the application. After considering these factors along with the

[2] Since zero tail probability equals to $-\infty$ in the logarithmic scale, we set an artificial lower bound of $\mathcal{T}$ based on the total number of simulation runs $N = 10^4$.

inflation rate of adopted pre-code, we decided to set the maximum acceptable decoding overhead $\hat{\varepsilon}_{max}$ of the SLLT post-code to 10 %. Consequently, we translated the above defined performance objectives into two target points and assigned to each of them one optimization goal [Specification 2].

6.4.4 Weight assignment

The performance requirements imposed on the SLLT code are often competing and usually cannot be achieved simultaneously. The strength, with which will be the SLLT code performance surface pulled towards a specific target point, should be expressed through the weight of corresponding goal(s). The higher importance the goal has (with respect to the ultimate purpose of the application), the higher weight the goal should have assigned. Since in our design exercise we consider attaining both target points equally important, we assigned equal weights to both of them [Specification 2].

6.5 Performance evaluation of designed code

In Fig. 4, we compare the decoding performance of three candidate SLLT codes that may be used as post-codes of the short-length raptor code: (a) SLLT code with minimized average overhead yielding the lossless decoding; (b) SLLT code with minimized average failure rate at 10 % of decoding overhead; and (c) SLLT code specially customized to match the erasure recovery capabilities of the chosen LDPC pre-code according to the Specification 2. It can be seen that all three SLLT codes exhibit very different degree distributions and performance profiles. However, since the chosen LDPC pre-code exhibits the ability to recover all the erased symbols as long as the erasure rate is below 5 %, the decoding performance of all three corresponding raptor codes (Fig. 5) is almost identical in this region of the failure rate. On the other hand, the performance profiles of all three raptor codes in the region above 5 % of the decoding failure rate are almost identical with the performance profiles of their SLLT post-codes, since in this region the ability of the pre-code to recover the erased symbols is only very limited. Based on the fact that the SLLT code specially customized to match the erasure recovery capabilities of the LDPC pre-code possesses the smallest portion of non-zero probability values in the region where the pre-code is unable to recover the erased symbols, we may conclude that the raptor code employing this post-code outperforms the other two candidate raptor codes. Our claim supports also the fact that the raptor code employing the

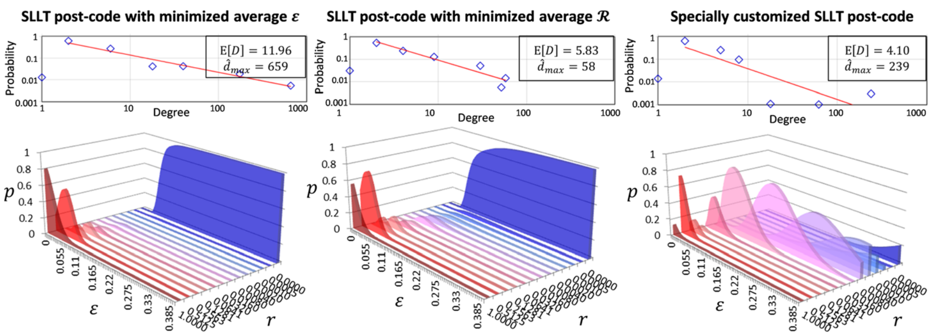


Fig. 4 Degree distribution and decoding performance comparison of three candidate SLLT post-codes

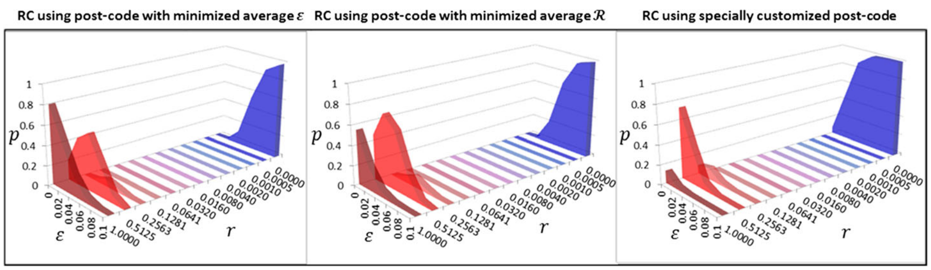


Fig. 5 Decoding performance comparison of three raptor codes employing different SLLT post-codes

specially customized SLLT post-code significantly outperforms the other two raptor codes in all three video benchmarks used in our experiments (Figs. 6 and 7).

7 Customizing decoding performance profile of SLLT codes in region of interest

7.1 Gradually improving decoding performance

There are various applications that require erasure protection codes with specially tapered decoding performance profiles fine-tuned throughout the specific range of decoding overhead, and hence, the SLLT codes with the steepest improvement of decoding performance may not be the most appropriate candidates for this class of applications. On the contrary, the performance objectives of SLLT codes designed to provide protection for these applications should aim at controlling the probability distribution of their decoding failure rates throughout the admissible range of decoding overhead. One important class of applications that may require SLLT codes with gradually improving erasure protection is unequal erasure protection (UEP) of the H.264 SVC multicast, where the playback quality of the SVC/MGS bitstream should improve gradually over a wide range of decoding overhead, since rapid improvement of the playback quality (causing the SVC bitstream to be either totally undecodable or completely decodable) may render the function of the UEP ineffective.

7.1.1 Region of interest

Region of interest is a region in the performance space determined by the admissible range of decoding overhead, in which the failure rate $\mathcal{R}$ and its tail probability $\mathcal{T}$ should drop gradually and

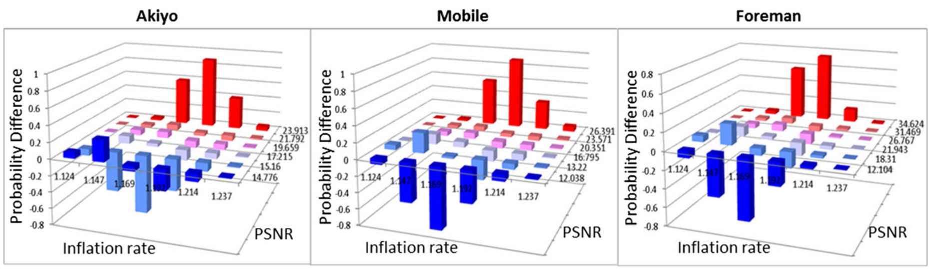


Fig. 6 Difference between PSNR values of raptor code employing post-code (c) and raptor code employing post-code (a)

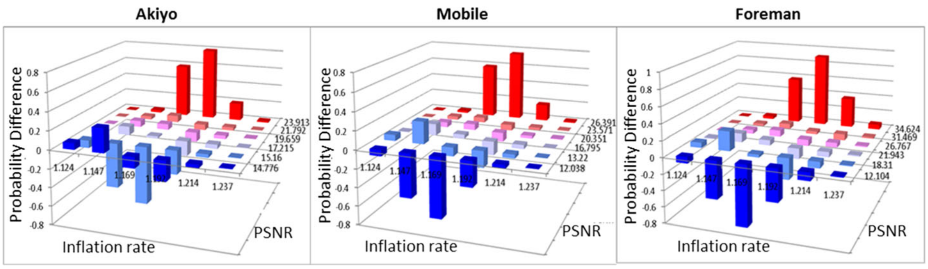


Fig. 7 Difference between PSNR values of raptor code employing post-code (c) and raptor code employing post-code (b)

specific decoding performance should be delivered at its lower and upper boundary. Based on these facts, the region of interest can be represented as a box in the $(\varepsilon, \mathcal{R}, \mathcal{T})$-space, whereas, the vertices of this box are determined by the desirable values of the performance parameters (i.e., $\hat{\varepsilon}_{min}, \hat{\mathcal{R}}_{min}, \hat{\mathcal{T}}_{min}$) and their maximum admissible counterparts (i.e., $\hat{\varepsilon}_{max}, \hat{\mathcal{R}}_{max}, \hat{\mathcal{T}}_{max}$). Ideally, the actual performance surface of the SLLT code attains the diagonal vertices representing the target performance characteristics (i.e., $P_1 \triangleq \left[\hat{\varepsilon}_{min} \ \ \hat{\mathcal{R}}_{max} \ \ \hat{\mathcal{T}}_{max}\right]$ and $P_2 \triangleq \left[\hat{\varepsilon}_{max} \ \ \hat{\mathcal{R}}_{min} \ \ \hat{\mathcal{T}}_{min}\right]$) and gradually improves throughout the defined box. Note that point P_1 corresponds to the minimum tolerable decoding quality at specific low decoding overhead, while point P_2 represents enhanced decoding quality at maximum tolerable level of decoding overhead.

7.1.2 Goal setting

In order to deliver desired gradually improving decoding performance profile, specific performance characteristics must be first provided at the lower and upper boundary of the admissible range of decoding overhead. In order to achieve the desired decoding performance (i.e., minimum tolerable decoding quality: $\hat{\mathcal{R}}_{max}, \hat{\mathcal{T}}_{max}$) at the lower boundary of this range $\hat{\varepsilon}_{min}$, the design goal(s) should take any of the following forms:

a) Minimize the positive distance between the actual and target value of decoding overhead $]\varepsilon - \hat{\varepsilon}_{min}]$ yielding minimum tolerable decoding quality $\left(\hat{\mathcal{R}}_{max}, \hat{\mathcal{T}}_{max}\right)$.
b) Minimize the positive distance between the actual and target value of symbol decoding failure rate $]\mathcal{R} - \hat{\mathcal{R}}_{max}]$ yielding tail probability $\hat{\mathcal{T}}_{max}$ at lower boundary of admissible region of decoding overhead $\hat{\varepsilon}_{min}$.
c) Minimize the positive distance between the actual and target value of tail probability $]\mathcal{T} - \hat{\mathcal{T}}_{max}]$ at failure rate $\hat{\mathcal{R}}_{max}$ and lower boundary of admissible region of decoding overhead $\hat{\varepsilon}_{min}$.

Similarly, in order to achieve the desirable decoding performance (i.e., enhanced decoding quality: $\hat{\mathcal{R}}_{min}, \hat{\mathcal{T}}_{min}$) at the upper boundary of the admissible range of decoding overhead $\hat{\varepsilon}_{max}$, the design goal(s) should take any of the following forms:

a) Minimize the positive distance between the actual and target value of decoding overhead $]\varepsilon - \hat{\varepsilon}_{max}]$ yielding specific enhanced decoding quality $\left(\hat{\mathcal{R}}_{min}, \hat{\mathcal{T}}_{min}\right)$.

b) Minimize the positive distance between the actual and target value of symbol decoding failure rate $]\mathcal{R}-\hat{\mathcal{R}}_{min}]$ yielding specific low tail probability $\hat{\mathcal{T}}_{min}$ at upper boundary of admissible region of decoding overhead $\hat{\varepsilon}_{max}$.
c) Minimize the positive distance between the actual and target value of tail probability $]\mathcal{T}-\hat{\mathcal{T}}_{min}]$ at specific low failure rate $\hat{\mathcal{R}}_{min}$ and upper boundary of admissible region of decoding overhead $\hat{\varepsilon}_{max}$.

7.2 Procedure for developing goal programs enabling search for SLLT codes with gradually improving decoding performance capable to protect multimedia source codes

As mentioned in previous section, the decoding performance of multimedia source codes is usually quantified in terms of their rate-distortion profiles that differ significantly from the (ε, $\mathcal{R}$, $\mathcal{T}$) performance model of the SLLT codes. Therefore, we devised a five-step procedure that we suggest to follow in order to develop goal programs capable to find SLLT codes providing protection to multimedia source codes with most appropriate gradually improving decoding performance:

a) Identify the performance profiles of the source code and/or pre-code: As described in Section 6.2.
b) Define the region with expected gradually improving decoding performance (region of interest): Specify the admissible range of each performance parameter (ε, $\mathcal{R}$, $\mathcal{T}$) based on the real-time requirements of given multimedia application, rate-distortion characteristics of the source data, erasure recovery capabilities of adopted pre-code and playback PSNR probabilities of protected bitstreams.
c) Formulate performance objectives of the SLLT code: Identify the key vertices of the region with expected gradually improving decoding performance representing the desired performance characteristics and choose the axes, along which the distance to these vertices should be minimized.
d) Translate performance objectives into goal program blocks: As described in Section 6.2.
e) Assign weights to individual goals according to their priorities: As described in Section 6.2.

In this section, we use the design of an SLLT post-code for rateless UEP code as an example to demonstrate the way of carrying out this procedure.

7.3 Protecting SVC multicast

In the *wireless SVC multicasting* over dynamic lossy communication channels, the symbol loss rates experienced by individual receivers may vary drastically over the space and time, and therefore, the rateless universal codes may in this case provide more robust erasure protection than conventional deterministic codes. Moreover, it is important to note that scalable data streams exhibit incremental cost-benefit profiles, for example, in case of the multimedia streaming they have gradually rising the *rate-distortion* curves. As consequence, it is beneficial to provide stronger protection to the *base layer* as well as to the *I* and *P frames*, on which the successful decoding of the entire *group of pictures* (GOP) depends, and offer only weaker protection to the incremental *medium grain scalable* (MGS) layers of the bitstream. Based on these facts, the employment of the UEP code with proper number of protection layers and sufficient erasure correction capabilities may offer gradual improvement of the multimedia playback quality starting at much lower decoding overhead than usual *forward erasure protection* (FEC). In subsequent sections, we illustrate the employment of proposed approach together with specified procedure for developing goal programs through the design of SLLT post-code that enables the rateless UEP code to support graceful degradation of the H.264 SVC playback quality under heavy packet loss.

7.4 Design of SLLT post-code for rateless UEP code protecting SVC multicast

Specification 3 Goal Program Setup to Protect SVC Multicast

```
$\mathcal{J} = 2$
Performance_Block$_1$ {
        $Target = (0.04, -5.06, -1.5)$                    // $\hat{r} = 0.03$ and $\hat{T} = 0.0316$
        $Goals = 2$
        $(g_1, w_1) = (\varepsilon, 0.05)$
        $(g_2, w_2) = (\mathcal{R}, 0.05)$
}
Performance_Block$_2$ {
        $Target = (0.08, -6.64, -1.5)$                    // $\hat{r} = 0.01$ and $\hat{T} = 0.0316$
        $Goals = 1$
        $(g_1, w_1) = (\mathcal{T}, 0.9)$
}
$Norm\_Degree = \infty$
```

7.4.1 Performance profiles of source data and pre-code

From the rate-distortion shown in Fig. 8 can be seen that *mean square error* (MSE) of the SVC/MGS groups of pictures often settles at few discrete levels, which means that few UEP layers with compatible coding rates [26] are sufficient to protect the multimedia bitstreams. Based on this fact, we chose to protect the bitstreams with UEP pre-code that possesses three SVC/MGS layers with 1:1:1 ratio among the bit rates and proper erasure protection. More precisely, we adopted as pre-code a three-layered *UEP convolutional code* [36] because it provides a code designer capability to cope with variations of the input stream size, although, it

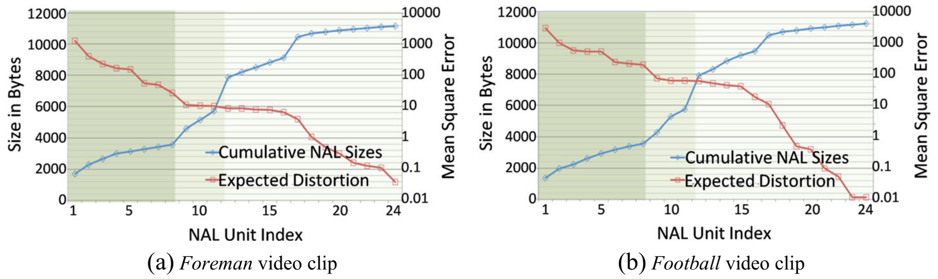


Fig. 8 Sizes (*blue*) and expected distortion ratios (*red*) of different NAL units in SVC-MGS Groups of Pictures

Table 3 Specification of adopted UEP convolutional pre-code

$\mathcal{N}$	$\mathcal{K}$	Generator matrix	Puncturing matrix	Separation vectors
4	3	$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 2 & 3 & 1 & 0 \\ 57 & 135 & 100 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	Original Code: 2 4 9 Punctured Code: 2 3 5

suffers from a minor sacrifice of the overall coding rate. In order to reduce the inflation rate of the composite code, we decided to raise the code rate of the pre-code through *puncturing* providing the maximum separation vector. The puncturing specification adopted in our design is reported in Table 3, where $\mathcal{K}$ and $\mathcal{N}$ represent the number of input and output sequences, respectively. Figure 9 shows the decoding performance of adopted pre-code in the random erasure channel. For more information about the design of the adopted UEP convolutional pre-code, an interested reader is referred to [42].

7.4.2 Region with expected gradually improving decoding performance

The main goals of the rateless UEP code designed to protect SVC multicast are to delivery: 1) decent decoding quality of the base layer at the lower boundary of the admissible range of overall inflation rate; and 2) practically full decoding quality of all the SVC/MGS layers at the upper boundary of this range. As consequence, the region of interest with respect to the decoding performance of the SLLT post-code can be specified by eight vertices that have coordinates determined by all the possible combinations of following terms: lower and upper boundary of the admissible range of decoding overhead ($\hat{\varepsilon}_{min}$ and $\hat{\varepsilon}_{max}$), decoding performance determining decent decoding quality of base layer ($\hat{\mathcal{R}}_{max}$ and $\hat{\mathcal{T}}_{max}$) and decoding performance determining full decoding quality of all three SVC/MSG layers ($\hat{\mathcal{R}}_{min}$ and $\hat{\mathcal{T}}_{min}$). Note that the target points $P_1 \triangleq \left[\hat{\varepsilon}_{min} \ \hat{\mathcal{R}}_{max} \ \hat{\mathcal{T}}_{max}\right]$ and $P_2 \triangleq \left[\hat{\varepsilon}_{max} \ \hat{\mathcal{R}}_{min} \ \hat{\mathcal{T}}_{min}\right]$ are the key diagonal

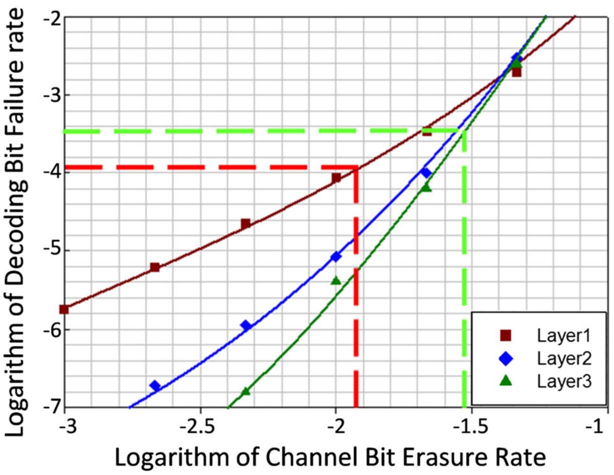


Fig. 9 Bit failure rate of adopted UEP convolutional pre-code with 3 SVC/MGS layers

vertices of the region of interest that represent the trend, in which the decoding performance of the SLLT post-code should exhibit gradual improvement.

7.4.3 Performance objectives

Based on the above mentioned facts, we decided to impose on designed SLLT post-code following performance objectives:

- Minimize the positive distance between actual failure rate $\mathcal{R}$ and its admissible upper boundary $\hat{\mathcal{R}}_{max}$ yielding tail probability $\hat{\mathcal{T}}_{max}$ enabling the pre-code to decode significant portion of the base layer at the lower boundary of admissible range of decoding overhead $\hat{\varepsilon}_{min}$;
- Minimize the positive distance between actual decoding overhead ε and its admissible lower boundary $\hat{\varepsilon}_{min}$ yielding the failure rate $\hat{\mathcal{R}}_{max}$ with tail probability $\hat{\mathcal{T}}_{max}$ enabling the pre-code to decode significant portion of the base layer;
- Minimize the positive distance between actual tail probability $\mathcal{T}$ and its admissible lower boundary $\hat{\mathcal{T}}_{min}$ at the failure rate $\hat{\mathcal{R}}_{min}$ enabling the pre-code to approach full decoding quality after receiving the upper bound of admissible range of decoding overhead $\hat{\varepsilon}_{max}$.

7.4.4 Target points and optimization goals

In order to specify the coordinates of target points corresponding to the defined performance objectives, it is necessary to find out the values of following thresholds:

- Decoding overhead $\hat{\varepsilon}_{min}$ determining the lower boundary of admissible range of decoding overhead;
- Decoding overhead $\hat{\varepsilon}_{max}$ determining the upper boundary of admissible range of decoding overhead;
- Failure rate $\hat{\mathcal{R}}_{max}$ and tail probability $\hat{\mathcal{T}}_{max}$ determining the successful decoding of significant portion of the base layer;
- Failure rate $\hat{\mathcal{R}}_{min}$ and tail probability $\hat{\mathcal{T}}_{min}$ determining practically complete decoding of all three SVC/MGS layers.

Since the starting point of transitional "waterfall" region of the SLLT code performance usually indicates the minimum level of decoding overhead, at which significant portion of the symbols can be successfully recovered, we decided to use its estimated lower bound as the lower boundary of the admissible range of decoding overhead. Based on our experience, if the block length is short ($10^3 \leq K \leq 10^4$ symbols), the transitional performance region of well-performing SLLT codes usually starts after receiving 0.04 extra symbols, and therefore, we set 0.04 as the lower boundary for the admissible range of decoding overhead $\hat{\varepsilon}_{min}$. On the other hand, the upper boundary of this range can be derived from the upper limit of the inflation rate normally permitted in the video multicast. Since the tolerable upper limit of the inflation rate in the video multicast is usually around 20 %, which in our case corresponds to 8 % of the SLLT post-code overhead, we set this value as the upper boundary of the admissible range $\hat{\varepsilon}_{max}$.

In order to find out the boundaries of the region of interest with respect to the decoding failure rate, it is necessary to analyze the erasure correction capabilities of adopted pre-code because the overall decoding performance of the rateless UEP code can be estimated by

Table 4 PSNR improvement in the base layer (mean)

Number of erased symbols	*T*				
	1.00E-1	3.16E-2	1.00E-2	3.16E-3	1.00E-3
3	0.00E+0	0.00E+0	9.70E-2	9.70E-2	9.70E-2
2	1.86E-1	2.20E-1	2.20E-1	2.20E-1	2.20E-1
1	2.20E-1	5.06E-1	5.21E-1	5.21E-1	5.21E-1
0	1.00E+0	1.00E+0	1.00E+0	1.00E+0	1.00E+0

looking up the *bit failure rates* of its pre-code at the *channel bit erasure rate* corresponding to the decoding failure rate of its post-code (Fig. 9). In [35] author shows that when employing the adopted UEP convolutional code, the bit failure rates enabling to decode significant portion of the base layer and approach the complete decoding of all three SVC/MGS layers are $3.6 \cdot 10^{-4}$ and $1.2 \cdot 10^{-4}$, respectively. Based on this fact, we set the admissible SLLT post-code failure rate boundaries to $\hat{\mathcal{R}}_{max} = 3{\cdot}10^{-2}$ and $\hat{\mathcal{R}}_{min} = 10^{-2}$ (Fig. 9).

On the other hand, in order to determine the corresponding boundaries with respect to the tail probability, we believe that it is reasonable to use the PSNR improvement data as shown in Tables 4 and 5. Note that we generated the content of these tables from 10^5 simulation runs of the *Foreman video sequence* based on the fact that this video clip represents a balanced mixture of the fine texture and fast motion. We consider appropriate to choose the column of Table 4 containing item(s) from interval [0.5, 0.7] as the upper boundary for the admissible range of tail probability and the column of Table 5 containing item(s) from interval [0.9, 1] as its lower boundary. Since there is more than one column in each table that contains such items, we decided to choose columns where these items possess the steepest improvement with respect to the tail probability. An interested reader can see that items with mentioned characteristics reside in both tables at $T = 0.0316$, and hence, we set both $\hat{\mathcal{T}}_{min}$ and $\hat{\mathcal{T}}_{max}$ to −1.5.

Based on the above mentioned facts, we translated the performance objectives imposed on the decoding performance of the SLLT post-code into two target points and three goals as shown in Specification 3.

7.4.5 Weight assignment

Failing to provide decent decoding quality of the base layer at the lower boundary of the admissible range of decoding overhead may be usually tolerable, however, failing to approach

Table 5 PSNR improvement in the enhancement layer 2 (mean)

Number of erased symbols	*T*				
	1.00E-1	3.16E-2	1.00E-2	3.16E-3	1.00E-3
3	5.53E-1	6.00E-1	6.13E-1	6.18E-1	6.19E-1
2	6.38E-1	6.94E-1	7.11E-1	7.17E-1	7.18E-1
1	7.29E-1	7.92E-1	8.12E-1	8.18E-1	8.21E-1
0	8.82E-1	9.62E-1	9.88E-1	9.96E-1	9.99E-1

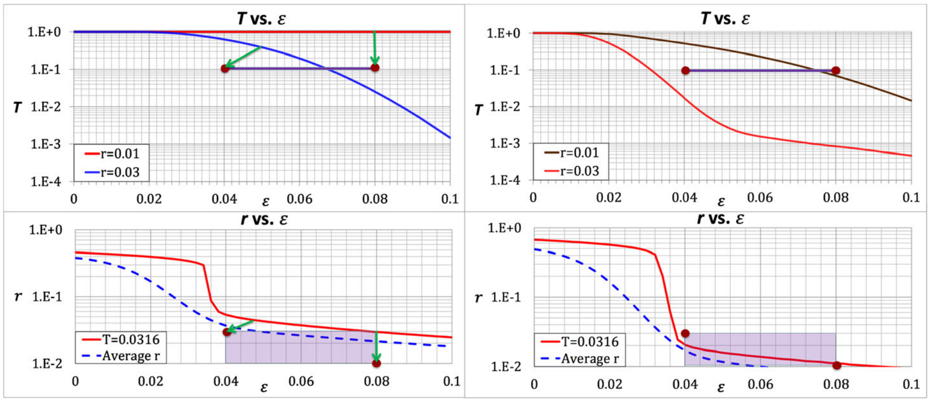


Fig. 10 Decoding performance of two SLLT post-codes showing "region of interest" (conventionally optimized SLLT post-code on the left and specially customized SLLT post-code on the right)

the full decoding quality at the upper boundary of this range will most likely render the composite rateless UEP code impractical for the protection of SVC multicast. Based on this fact, attaining the target point of Performance_Block$_2$ should be a priority of our design exercise, and therefore, we assigned a dominant weight to the corresponding goal [Specification 3].

7.5 Performance evaluation of designed code

In order to demonstrate the capabilities of proposed approach, we compare in this subsection the decoding performance of two rateless UEP codes designed by concatenating the chosen UEP convolutional pre-code with two different SLLT post-codes: (a) SLLT post-code with minimized average failure rate at the lower boundary of the range of admissible decoding overhead (i.e., at $\hat{\varepsilon} = 0.04$); and (b) SLLT post-code specially customized according to the Specification 3 to match the UEP capabilities of adopted pre-code. Note that as the maximum symbol size and block length K of the customized SLLT post-code, we chose 32 bits and 6000 symbols, respectively.

Figure 10 shows the decoding performance of both candidate post-codes, whereas, the purple line/rectangle represents the admissible region, in which the decoding performance of the SLLT code should exhibit the gradual improvement, and the brown filled circles represent the key vertices (i.e., target points) of this region.

Since the curves of the specially customized SLLT post-code representing $r = 0.01$, $r = 0.03$ and $T = 0.0316$ succeed to reach (and even slightly over-satisfy) the corresponding target points,

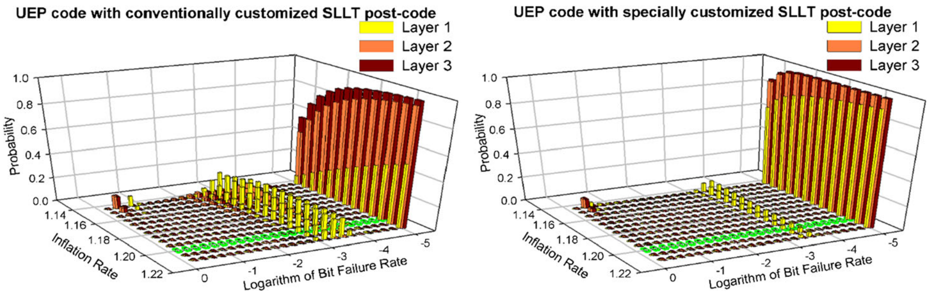


Fig. 11 Bit failure rates of designed rateless UEP codes

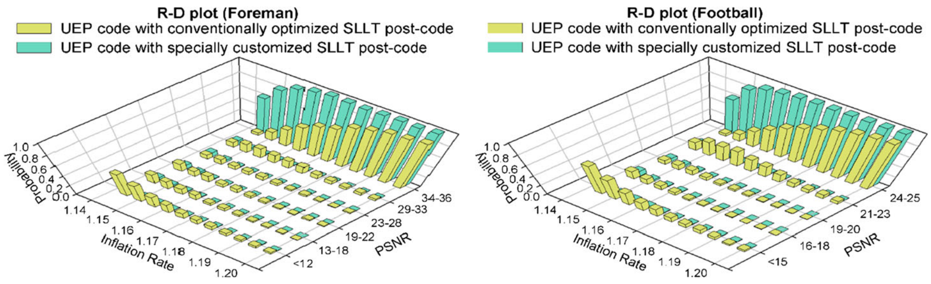


Fig. 12 Playback PSNR probabilities of SVC/MGS bitstreams protected by designed rateless UEP codes

it can be concluded that the rateless UEP code that concatenates this post-code with above described convolutional UEP pre-code exhibits the ability to successfully decode the significant portion of the base layer at the lower boundary of the admissible range of the decoding overhead as well as deliver practically complete decoding of all the three SVC/MSG layers at its upper boundary. On the other hand, since the corresponding curves of the SLLT post-code with minimized average failure rate fails to reach their respective target points, this code does not leave sufficient room for the convolutional pre-code to fully leverage from its UEP capability within the admissible range of the decoding overhead. Note that length of the arrows that point to the key vertices of the admissible region (Fig. 10) can be seen as the distance between the actual and target performance of the conventionally optimized SLLT post-code. In addition, the superiority of the rateless UEP code employing the specially customized SLLT post-code with respect to the graceful degradation is clearly shown in Figs. 11 and 12.

8 Conclusions and future work

In this paper, we present simulation-based design approach that adapts novel statistical decoding performance model along with the distance based goal programming and state-of-the-art stochastic optimization method CMA-ES in order to search for customized SLLT codes satisfying specific decoding performance requirements. Furthermore, this paper contains general rules how to use proposed approach and illustrates them on two real-world examples, namely, on the design of (1) SLLT post-code of the raptor code, specially customized w.r.t. two competing performance objectives located in different regions of decoding performance; and (2) SLLT post-code of the rateless UEP code that has decoding performance fine-tuned w.r.t. the rate-distortion performance of the SVC bitstream in the region of admissible decoding overhead. Results reported in Sections 6–7 demonstrate that as long as the performance objectives imposed on the SLLT code are properly derived from the rate-distortion characteristics of the source data and/or erasure recovery capabilities of the pre-code, and match the performance requirements of the application, the proposed approach can be used to generate SLLT codes with desirable decoding performance.

In the future, we plan to focus on: (1) modifications of the randomized encoding input selection schemes in order to further improve the SLLT code performance; (2) implementation of one of the variance reduction techniques into the SLLT code fitness evaluation in order to reduce the noise caused by employed Monte-Carlo simulations; and (3) application of the proposed customization approach to the domain of medical data transfer protection.

Acknowledgements The authors would like to thank Prof. Chung-Hsuan Wang of the NCTU Electrical Engineering Department for his valuable advices and fruitful discussion on SLLT codes. We are also grateful to Pei-lun Diao for a major contribution to the design of statistical decoding performance model. This work was supported in part by the Aiming for the Top University Plan of National Chiao Tung University, sponsored by the Ministry of Education, Taiwan, under Grant Number 105W963, in part by the Ministry of Science and Technology, Taiwan, under Grant no. MOST 105-2923-E-009 -001 -MY2.

References

1. (2004) 3GPP technical specification for multimedia broadcast/multicast service: protocols and codecs. 3GPP TS26.346
2. (2010) RaptorQ ^{TM} Technical Overview
3. Ahmad S, Hamzaoui R, Al-Akaidi MM (2011) Unequal error protection using fountain codes with applications to video communication. IEEE Trans Multimedia 13(1):92–101
4. Belton V (ed) (2002) Goal and reference point methods. In Multiple Criteria Decision Analysis, no. 1986, Kluwer Academic Publishers, pp 209–232
5. Bodine EA, Cheng MK (2008) Characterization of luby transform codes with small message size for low-latency decoding. In IEEE International Conference on Communications, pp 1195–1199
6. Chen C-M, Chen Y (2012) Sparse degrees analysis for LT codes optimization. 2012 I.E. Congr. Evol. Comput., pp 1–6
7. CMA-ES. [Online]. Available: http://en.wikipedia.org/wiki/CMA-ES
8. Emmerich M, Deutz A (2006) Multicriteria optimization and decision making. Leiden University
9. Hansen N, Kern S (2004) Evaluating the CMA evolution strategy on multimodal test functions. In Parallel Problem Solving from Nature - PPSN VIII, Springer Berlin Heidelberg, pp 282–291
10. Hansen N, Ostermeier A (1996) Adapting arbitrary normal mutation distribution in evolution strategies - the covariance matrix adaptation. In IEEE International Conference on Evolutionary Computation, pp 312–317
11. Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evol Comput 9(2):159–195
12. Husna J (2015) Customization of SLLT code performance using multi-objective evolutionary computation methods. National Chiao Tung University
13. Hyytiä E, Tirronen T, Virtamo J (2006) Optimizing the degree distribution of LT codes with an importance sampling approach. In 6th International Workshop on Rare Event Simulation, pp 64–73
14. Ignizio JP, Romero C (2003) Goal programming. Encyclopedia of Information Systems, vol. 2. Elsevier, pp 489–500
15. Joines JA, Houck CR (1994) On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. In IEEE World Congress on Computational Intelligence, pp 579–584
16. Lin JG (2005) On min-norm and min-max methods of multi-objective optimization. Math Program 103:1–33
17. Liva G, De Cola T, Paolini E, Chiani M, Calzolari GP (2008) Packet loss recovery via maximum-likelihood LDPC decoding. CCSDS Fall Meeting, Berlin
18. Luby M (2002) LT codes. 43rd Annu. IEEE Symp. Found. Comput. Sci. Proceedings., pp 271–280
19. Luby M, Gasiba T, Stockhammer T, Watson M (2007) Reliable multimedia download delivery in cellular broadcast networks. IEEE Trans Broadcast 53(1):235–246
20. Ma Y, Xu J, Tang J, Long K (2010) Truncated robust soliton distribution on LT codes for multimedia transmission. In International Conference on Multimedia Technology, pp 1–4
21. Maatouk G, Shokrollahi A (2012) Analysis of the second moment of the LT decoder. IEEE Trans Inf Theory 58(5):2558–2569
22. Mackay DJC Encyclopedia of sparse graph codes. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/da
23. Marler RT, Arora JS (2004) Survey of multi-objective optimization methods for engineering. Struct Multidiscip Optim 26(6):369–395

24. Multi-objective optimization. [Online]. Available: http://en.wikipedia.org/wiki/Multi-objective_optimization
25. Oka A, Lampe L (2009) Data extraction from wireless sensor networks using distributed fountain codes. IEEE Trans Commun 57(9):2607–2618
26. Shokrollahi A (2004) LDPC codes: an introduction. In Coding, Cryptography and Combinatorics, Birkhäuser Basel, pp 85–110
27. Shokrollahi A (2006) Raptor codes. IEEE Trans Inf Theory 52(6):2551–2567
28. Shokrollahi A, Luby M (2011) Raptor codes. Found Trends Commun Inf Theory 6(3–4):213–322
29. Shokrollahi A, Storn R (2000) Design of efficient erasure codes with differential evolution. In IEEE International Symposium on Information Theory, p 5
30. Sørensen JH, Popovski P, Østergaard J (2012) Design and analysis of LT codes with decreasing ripple size, pp 1–22
31. Steuer RE (1986) Multiple criteria optimization: theory, computation and applications. Wiley, New York
32. Storn R (1996) Differential evolution design of an IIR-filter. Proc. IEEE Int. Conf. Evol. Comput.
33. Talari A, Rahnavard N (2012) On the intermediate symbol recovery rate of rateless codes. IEEE Trans Commun 60(5):1237–1242
34. Vukobratovic D, Stankovic V, Sejdinovic D, Stankovic L, Xiong Z (2009) Scalable video multicast using expanding window fountain codes. IEEE Trans Multimedia 11(6):1094–1104
35. Wang B-W (2014) Customization of short-length LT codes for rateless coding schemes used in wireless video multicasting. National Chiao Tung University
36. Wang C, Chiu M, Chao C (2010) On unequal error protection of convolutional codes from an algebraic perspective. IEEE Trans Inf Theory 56(1):296–315
37. Wang C-H, Zao JK, Chen H-M, Diao P-L, Chiu C-M (2011) A rateless UEP convolutional code for robust SVC/MGS wireless broadcasting. In IEEE International Symposium on Multimedia, pp 279–284
38. Yen K, Chen C, Chang H (2012) An improved LT encoding scheme with extended chain lengths. In International Symposium on Information Theory and its Applications, pp 11–15
39. Yen K, Liao Y, Chen C, Zao JK, Chang H (2013) Integrating non-repetitive LT encoders with modified distribution to achieve unequal erasure protection. IEEE Trans Multimedia 15(8):2162–2175
40. Yen K, Liao Y, Chen C, Chang H (2013) Modified robust soliton distribution (MRSD) with improved ripple size for LT codes. IEEE Commun Lett 5:976–979
41. Yen K-K, Liao Y-C, Chen C-L, Chang H-C (2013) Adjusted robust soliton distribution (ARSD) with reshaped ripple size for LT codes. In International Conference on Wireless Communications and Signal Processing (WCSP), pp 1–6
42. Zao JK, Hornansky M, Diao P-L, Wang B-W, Wang C-H, Chang L-J (2012) Optimized rateless UEP codes for scalable video streaming. In 2012 Visual Communications and Image Processing, pp 1–6
43. Zhu H, Xie Z (2008) Advanced LT codes in satellite data broadcasting system. 2008 11th IEEE Singapore Int. Conf. Commun. Syst., pp 1431–1435
44. Zhu H, Zhang C, Lu J (2007) Designing of fountain codes with short code-length. In 3rd International Workshop on Signal Design and Its Applications in Communications, pp 65–68
45. Zielinski K, Peters D, Laur R (2008) Stopping criteria for single-objective optimization. In Advances in Differential Evolution, no. 1, Springer Berlin Heidelberg, pp 111–138

Martin Hornansky received double master degree from Martin Luther University in Halle-Wittenberg and University of Economics in Bratislava as well as M.Sc. degree from Comenius University. Currently, he is pursuing a Ph.D. degree at the Institute of Computer Science and Engineering of the National Chiao Tung

University (NCTU) in Hsinchu. The main areas of his current research interest include stochastic optimization, evolutionary algorithms and video coding.

Dr. John K. Zao is currently an associate professor in the Computer Science Department of the National Chiao Tung University (NCTU) in Hsinchu, Taiwan, R.O.C. and the Co-chairman of the Security Workgroup in OpenFog Consortium. He received his B.A.Sc. in Engineering Science and M.A.Sc. in Electrical Engineering from University of Toronto and later S.M. and Ph.D. in Computer Science from Harvard University. He was elected a Senior Member of IEEE in 2001.

Since 2010, Dr. Zao has been collaborating with the Co-directors, Dr. Tzyy-Ping Jung and Dr. Scott Makeig, of the Swartz Center for Computational Neuroscience (SCCN) at the University of California in San Diego (UCSD) to develop real-world applications of augmented brain-computer interaction (A-BCI) with the aid of Fog and Cloud Computing. His current research interests include: Multi-tier Fog-Cloud Computing Architecture for IoT Applications; Efficient and Resilient H.264 SVC Heterogeneous Multicasting; Pervasive Telemonitoring and Mobile Healthcare.