# On the Intermediate Symbol Recovery Rate of Rateless Codes

Ali Talari, *Student Member, IEEE*, and Nazanin Rahnavard, *Member, IEEE*

***Abstract*—Existing *rateless* codes have low *intermediate symbol recovery rates* (ISRR). Therefore, we first design *new* rateless codes with close to optimal ISRR employing *genetic algorithms*. Next, we assume an estimate of the channel erasure rate is available and propose an algorithm to *further* improve the ISRR of the designed codes.**

***Index Terms*—Codes optimization, genetic algorithms, intermediate performance, rateless codes.**

## I. Introduction

RATELESS codes are modern *forward-error-correction* (FEC) codes with capacity-achieving performance on erasure channels [1]–[3]. A rateless encoder at a source $S$ can potentially generate a limitless number of output symbols $c_i, i \in \{1, 2, \ldots\}$ from $k$ input symbols $\mathbf{x} = \{x_1, x_2, \ldots, x_k\}$. Let $k\gamma$ and $z \in [0, 1]$ denote the number of received output symbols and the fraction of decoded input symbols, respectively, at a decoder $D$.

To generate an output symbol, first its degree is randomly chosen to be $d$ with probability $\Omega_d$ using a *probability distribution* $\{\Omega_1, \Omega_2, \ldots, \Omega_k\}$ (also shown by its generator polynomial $\Omega(x) = \sum_{i=1}^{k} \Omega_i x^i$). Next, $d$ input symbols are selected *uniformly at random* and are bitwise *XOR*ed to form the output symbol. We call the $d$ contributing input symbols in forming an output symbol as its *neighbors*. Rateless decoder *iteratively* finds output symbols with only one unrecovered neighboring input symbol and decodes their value by bitwise XOR operations. It has been shown that $D$ can fully decode $\mathbf{x}$ ($z = 1$) from *any* subset of $k\gamma_{succ}$ output symbols with high probability [1]–[3], where $\gamma_{succ}$ is called the *coding overhead* and is slightly larger than one.

Although rateless codes are capacity achieving, in *intermediate range*, i.e., $0 \leq \gamma \leq 1$, input symbols are barely decoded because most of the received output symbols are *buffered* for a later decoding [4]–[7]. Therefore, rateless codes have a low *intermediate symbol recovery rate* (ISRR), i.e., $z \approx 0$ at $0 \leq \gamma \leq 1$. However, in some applications such as multimedia content delivery *partial* recovery of the input symbols is still beneficial, which motivates us to design rateless codes with high ISRR.

Paper approved by S.-Y. Chung, the Editor for LDPC Coding and Information Theory of the IEEE Communications Society. Manuscript received January 12, 2011; revised September 13 and December 19, 2011.

This material is based upon work supported by the National Science Foundation under Grants ECCS-1056065 and CCF-0915994.

The authors are with the Department of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078 USA (e-mail: {ali.talari, nazanin.rahnavard}@okstate.edu).

Digital Object Identifier 10.1109/TCOMM.2012.030712.110032

In this paper, we first employ *multi-objective genetic algorithms* to design degree distributions that have *almost* optimal ISRR *throughout* $0 \leq \gamma \leq 1$. We employ the term "almost optimal" because genetic algorithms are known to find solutions that are not necessarily global-optimum but are rather very close to the global-optimum solution. Therefore, throughout our code design process the term optimal implies almost optimal. In the next step, we assume that an estimate of the channel erasure rate $\varepsilon \in [0, 1)$ is available at $S$ and propose *rateless coded symbol sorting* (RCSS), which rearranges the transmission order of output symbols to further improve the ISRR. Preliminary results of this paper have appeared in [8] and [9].

This paper is organized as follows. Section II provides a review on existing work. In Section III, we design distributions to realize high ISRR utilizing multi-objective genetic algorithms. In Section IV, we propose RCSS and discuss its capabilities. Finally, Section V concludes the paper.

## II. Related Work

In [4], the author shows that the intermediate range of rateless codes can be divided into three regions. The three intermediate regions for $0 \leq z < 1$ are $z \in [0, \frac{1}{2}]$, $z \in [\frac{1}{2}, \frac{2}{3}]$, and $z \in (\frac{2}{3}, 1)$, which approximately give the equivalent regions of $\gamma \in [0, 0.693]$, $\gamma \in [0.693, 0.824]$, and $\gamma \in [0.824, 1]$. Further, author designs optimal degree distributions that achieve the upper bound on ISRR of all rateless codes in these regions. However, the codes designed in [4] are asymptotically optimal and may not be employed when $k$ is finite. Further, the proposed degree distributions are only optimal in one intermediate region.

In [5], [7] authors propose to employ feedbacks from $D$ to keep $S$ aware of $z$. They propose to gradually increase the degree of output symbols such that the instantaneous recovery probability of each arriving output symbol is maximized. The codes designed in [5], [7] require feedbacks, hence their application is not always feasible.

Authors in [6] propose to transmit output symbols in the order of their *ascending degree*. Although this would increase the ISRR, we will see that RCSS always outperforms this technique.

## III. Rateless Code Design With High ISRR

In this section, we design degree distributions for rateless coding with various $k$'s employing multi-objective genetic algorithms.

### A. Decision Variables and Objective Functions

To obtain high ISRRs in all three intermediate regions, we need to tune the degree distribution $\Omega(.)$ considering all three intermediate regions of $0 \leq \gamma \leq 1$. We choose three overheads $\gamma = 0.5$, $\gamma = 0.75$, and $\gamma = 1$ (one from each intermediate region, see Section II) and define the respective value of $z$ at these $\gamma$'s as our objective functions. Let $z_{0.5,\Omega(.)}$, $z_{0.75,\Omega(.)}$, and $z_{1,\Omega(.)}$ denote the value of $z$ at three selected $\gamma$'s representing three objective functions that we aim to *concurrently maximize* and realize a high ISRR. With this setup, we have three *conflicting* objective functions meaning that improving $z$ at one point would decrease $z$ at one or both other $\gamma$'s. As a result, we employ *multi-objective optimization methods* to design our desired distributions.

Clearly, in our optimization problem the decision variables are entries of $\Omega(.)$. Codes that are designed to realize a high ISRR have $\Omega(.)$'s with much smaller *maximum* degree compared to codes designed for full input symbol recovery [1]–[3]. For instance, codes that optimally perform in the first and second intermediate regions have maximum degrees of only 1 and 2 [4], respectively. Consequently, we consider degree distributions with maximum degree of 50. Thus, we have fifty *decision variables* $\{\Omega_1, \Omega_2, \ldots, \Omega_{50}\}$ that take values in $[0, 1]$ such that $\sum_{i=1}^{50} \Omega_i = 1$. Later, we see that the optimum $\Omega(.)$'s have much smaller maximum degree than 50.

We need to take different approaches to find $z_{\gamma,\Omega(.)}$ for asymptotic and finite length setups. For asymptotic case, the expression providing the rateless decoding error rate has been previously obtained in [3], [10]–[12]. Let $v_l$ be the probability that an input symbol is not recovered after $l$ decoding iterations. From [3], [10]–[12], we have

$$v_l = \delta(1 - \beta(1 - v_{l-1})), l \geq 1 \tag{1}$$

in which $v_0 = 1$, $\beta(y) = \Omega'(y)/\Omega'(1)$, and $\delta(y) = e^{\Omega'(1)\gamma(y-1)}$. It can be shown that the sequence $\{v_l\}_l$ is convergent with respect to the number of decoding iterations, $l$ [11], [12]. Let $V_{\gamma,\Omega(.)}$ denote the corresponding fixed point. This fixed point is the final error rate of a rateless decoding with parameters $\Omega(.)$ and $\gamma$, hence $z_{\gamma,\Omega(.)} = 1 - V_{\gamma,\Omega(.)}$.

On the other hand, the expression for the error rate of rateless decoding for finite $k$ has been analyzed in [13], [14]. However, the high complexity of these expressions makes their application in genetic-algorithm implementation almost impossible. Therefore, to find $z$ for finite $k$ we employ Monte-Carlo method by averaging $z$ for a large enough number of decoding simulation experiments for $k \in \{10^2, 10^3, 10^4\}$. Similar to asymptotic case, our objective functions are $z_{0.5,\Omega(.)}$, $z_{0.75,\Omega(.)}$, and $z_{1,\Omega(.)}$, which in this case are found by numerical simulations.

### B. Optimized Rateless Codes for High ISRR

We employ NSGA-II multi-objective optimization algorithm [15] to find the distributions that have optimal $z$ at three selected $\gamma$'s (interested readers are encouraged to refer to [8, Section III.B] and [15] for more information on NSGA-II). The results of our optimizations are four *databases* of degree distributions optimized for $k \in \{10^2, 10^3, 10^4, \infty\}$. Due to huge size of the four databases they may not be reported in the paper and are made available online at [16]. In the next section, we investigate the performance of several designed distributions.

TABLE I
OPTIMUM DEGREE DISTRIBUTIONS FOR DIFFERENT WEIGHTS $\mathbf{W} = (W_{0.5}, W_{0.75}, W_1)$.

| $k$ | $\mathbf{W}$ | Optimum degree distribution $\Omega(y)$ |
|---|---|---|
| $10^2$ | $(1,1,1)$ | $0.348y + 0.652y^2$ |
| | $(0,1,0)$ | $0.1911y + 0.8082y^2 + 0.0003y^4$ |
| | $(0,0,1)$ | $0.116y + 0.467y^2 + 0.417y^3$ |
| | $(1,4,1)$ | $0.346y + 0.652y^2$ |
| | $(1,1,4)$ | $0.1515y + 0.7903y^2 + 0.0581y^3$ |
| $10^3$ | $(1,1,1)$ | $0.3131y + 0.6869y^2$ |
| | $(0,1,0)$ | $0.0139y + 0.9861y^2$ |
| | $(0,0,1)$ | $0.0624y + 0.5407y^2 + 0.2232y^4 + 0.1737y^5$ |
| | $(1,4,1)$ | $0.1448y + 0.8552y^2$ |
| | $(1,1,4)$ | $0.0624y + 0.9315y^2$ |
| $10^4$ | $(1,1,1)$ | $0.2474y + 0.7526y^2$ |
| | $(0,1,0)$ | $0.011y + 0.989y^2$ |
| | $(0,0,1)$ | $0.0312y + 0.4069y^2 + 0.3716y^3 + 0.0024y^6$ $+0.0264y^7 + 0.1519y^{10} + 0.0096y^{14}$ |
| | $(1,4,1)$ | $0.1452y + 0.8548y^2$ |
| | $(1,1,4)$ | $0.16y + 0.3524y^2 + 0.1318y^3 + 0.3553y^5$ $+0.0001y^7 + 0.0003y^{10} + 0.0001y^{14}$ |
| $\infty$ | $(1,1,1)$ | $0.29599y + 0.70401y^2$ |
| | $(0,1,0)$ | $0.00003y + 0.99997y^2$ |
| | $(0,0,1)$ | $0.00536y + 0.50088y^2 + 0.12547y^3$ $+0.17492y^4 + 0.03797y^5 + 0.00583y^6$ $+0.00011y^7 + 0.00013y^8 + 0.00001y^{10}$ $+0.00209y^{11} + 0.06425y^{13} + 0.08297y^{14}$ |
| | $(1,4,1)$ | $0.12469y + 0.87531y^2$ |
| | $(1,1,4)$ | $0.11003y + 0.24932y^2 + 0.34144y^3$ $+0.14488y^4 + 0.02164y^5 + 0.00123y^6$ $+0.00014y^{11} + 0.05257y^{13} + 0.07862y^{14}$ $+0.00012y^{17}$ |
| All | $(1,0,0)$ | $y$ |
| | $(4,1,1)$ | |

### C. Performance Evaluation of the Designed Codes

Based on the desired ISRR at each intermediate region an appropriate $\Omega(.)$ needs to be selected among the *many* optimum degree distributions in our databases. To facilitate the distribution selection from our databases we propose a *weighted* function $F(\Omega(.))$ defined by

$$\begin{aligned} F(\Omega(.)) =& W_{0.5}[Z_{0.5} - z_{0.5,\Omega(.)}] + W_{0.75}[Z_{0.75} - z_{0.75,\Omega(.)}] \\ &+ W_1[Z_1 - z_{1,,\Omega(.)}], \end{aligned} \tag{2}$$

where $Z_\gamma$ is the highest possible $z$ (upper bound on $z$) at $\gamma$ for all rateless codes and $W_\gamma$ is a *tunable weight*. From [4], we have $Z_{0.5} = 0.3934$, $Z_{0.75} = 0.5828$ and $Z_1 = 1$. For future references, we define $\mathbf{W} = (W_{0.5}, W_{0.75}, W_1)$. We can find $\Omega(.)$ of interest by setting the appropriate weights and selecting the $\Omega(.)$ that *minimizes* $F(\Omega(.))$. However, we emphasize that one may replace (2) with any desired linear or non-linear weighted function. Table I shows the optimum degree distributions for the selected arbitrary weights. Note that the degree distributions reported in Table I are *only samples* of many degree distributions we have made available at [16].

One may choose an optimal distribution based the desired weights from the databases provided at [16]. From Table I,

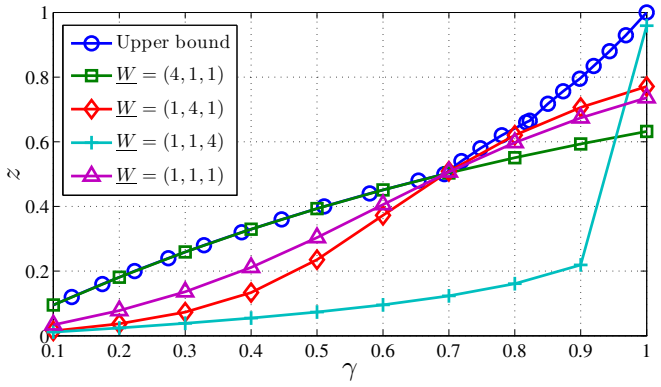


Fig. 1. ISRR of selected designed codes and the ISRR upper bound for asymptotic setup.

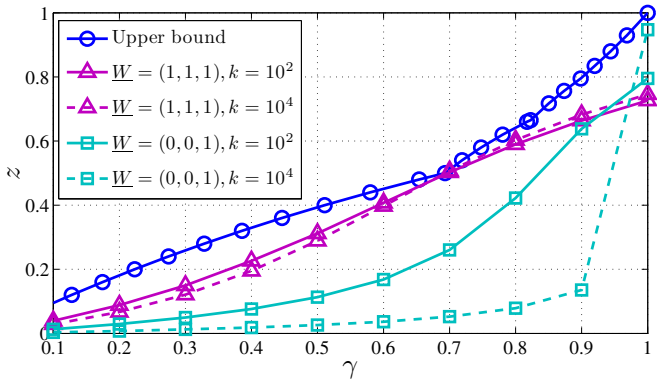


Fig. 2. ISRR of selected designed codes and the ISRR upper bound for $k = 10^2$ and $k = 10^4$.

we can see that the optimal degree distributions for finite length slightly differ from the distributions proposed in [4]. For instance for $\mathbf{W} = (0,1,0)$ the distribution is non-zero for $\Omega_1$, which allows the rateless decoding to start. Moreover, from our databases we observe that the maximum degree of all designed degree distributions is 19, which is much smaller than 50. Further, we can see that as $k$ decreases, large degrees are also eliminated. We compare the performance of our designed degree distributions with the upper bound found in [4] in Figs. 1 and 2.

The ISRR of the codes designed for $\mathbf{W} = (1,1,1)$ as are shown in Figs. 1 and 2 are optimal at three selected $\gamma$'s. In other words, there is *no other degree distribution* that can go closer to the upper bound at *one* $\gamma$ *without decreasing* $z$ for at least one other $\gamma$ compared to our designed degree distributions. Moreover, from Figs. 1 and 2 we can see that by setting the desired weights the selected distribution performs better at the region with the higher weight. Further, we can see that as $k$ increases the difference of ISRR with the upper bound decreases because the upper bound is derived for asymptotic setup. In the next section, we show how ISRR of our designed codes may be increased even more.

## IV. *RCSS*: Rateless Coded Symbol Sorting

In practice an estimate of the channel erasure rate $\varepsilon$ may be available at $S$ [17]. The value of $\varepsilon$ may be exploited as a side information to further improve the ISRR of rateless codes.

### A. RCSS: Rateless Symbol Sorting Algorithm

When $S$ has an estimate of $\varepsilon$, it is aware that in total $m = \frac{k\gamma_{succ}}{1-\varepsilon}$ output symbols should be transmitted so that $D$ receives $k\gamma_{succ}$ output symbols. The main idea in designing RCSS is that $S$ can generate $m$ output symbols ahead of transmission. Therefore, it can *rearrange* the order of $m$ output symbols such that each delivered symbol has the highest probability of decoding an input symbol at $D$. This results in a considerable improvement of ISRR since fewer output symbols are buffered for a later decoding at $D$. We should note that RCSS is merely implemented in $S$ and the decoder remains intact. Therefore, in contrast to [5], [6] we assume $D$ generates no feedback and RCSS can only employ the information available at $S$.

The reordering of $m$ output symbols in RCSS is performed as follows. $S$ maintains a probability vector $\boldsymbol{\rho} = [\rho(1), \rho(2), \ldots, \rho(k)]$, in which $\rho(j)$ represents the probability that $x_j$ is still *not recovered at* $D$. Clearly, $S$ initializes $\boldsymbol{\rho}$ to an all-one vector when the transmission has not started yet. At each transmission $S$ finds an output symbol $c_i$ that has the highest probability of recovering an input symbol at $D$ based on $\boldsymbol{\rho}$ (as described later). Next, $S$ transmits $c_i$ and updates $\rho(j), j \in \mathcal{N}(c_i)$, where $\mathcal{N}(c_i) \subset \{1,2,\ldots,k\}$ is a set containing index of input symbols that are neighboring to $c_i$. $S$ continues until all $m$ output symbols are transmitted.

From the rateless decoding procedure, we can see that an output symbol $c_i$ with degree $d$, i.e., $|\mathcal{N}(c_i)| = d$, where $|.|$ represents the cardinality of a set, can recover an input symbol $x_j$ iff all $x_w, w \in \{\mathcal{N}(c_i) - j\}$ have already been recovered. Let $p_{dec}(i), i \in \{1,2,\ldots,m\}$ denote the probability that $c_i$ can recover an input symbol at $D$. We assume $p_{dec}(i) = 0$ if $c_i$ has been previously transmitted. Since at the beginning of transmission no input symbol is still recovered, we have $p_{dec}(i) = 0$ if $|\mathcal{N}(c_i)| > 1$, i.e., output symbols with degrees larger than one cannot decode any input symbol at $D$. Besides, for $|\mathcal{N}(c_i)| = 1$ we have $p_{dec}(i) = (1-\varepsilon)$, i.e. only degree-one output symbols that are not erased on the channel (with probability $1-\varepsilon$) can recover an input symbol. Therefore, at the beginning of transmission degree-one output symbols have the highest probability of decoding an input symbol at $D$. Consequently, $S$ transmits degree-one $c_i$'s with $\mathcal{N}(c_i) = \{j\}$ and updates $\rho(j) = \varepsilon\rho_{old}(j)$, where $\rho_{old}(j)$ is the value of $\rho(j)$ before $c_i$ was transmitted.

Next, we consider a degree-two output symbol $c_i$ with $\mathcal{N}(c_i) = \{j,l\}$. In this case, $c_i$ can recover $x_j$ with probability $(1-\varepsilon)(1-\rho(l))\rho(j)$, which is the probability that $c_i$ is not dropped on channel, $x_j$ has not been recovered previously, and $x_l$ has already been recovered. Similarly, $c_i$ can recover $x_l$ with probability $(1-\varepsilon)(1-\rho(j))\rho(l)$. Consequently, $p_{dec}(i) = (1-\varepsilon)[(1-\rho(l))\rho(j)+(1-\rho(j))\rho(l)]$. Assume $\forall w \neq i, p_{dec}(i) > p_{dec}(w)$, i.e. $c_i$ has the highest probability of decoding an input symbol at $D$ among the remaining output symbols. Therefore, $S$ transmits $c_i$ next

and sets $\rho(j) = \rho_{old}(j)(1-(1-\varepsilon)(1-\rho_{old}(l)))$ and $\rho(l) = \rho_{old}(l)(1-(1-\varepsilon)(1-\rho_{old}(j)))$.

Further, we consider an output symbol $c_i$ with $|\mathcal{N}(c_i)| = d$. Such a $c_i$ can decode an $x_j, j \in |\mathcal{N}(c_i)|$ with probability $(1-\varepsilon)\rho(j) \prod_{v\in\mathcal{N}(c_i),v\neq j}(1-\rho(v))$. Therefore, $p_{dec}(i) = (1-\varepsilon)\sum_{l\in\mathcal{N}(c_i)}[\rho(l)\prod_{v\in\mathcal{N}(c_i),v\neq l}(1-\rho(v))]$. If $\forall w \neq i, p_{dec}(i) > p_{dec}(w)$, $S$ transmits $c_i$ and updates $\rho(j) = \rho_{old}(j)[1-(1-\varepsilon)\prod_{v\in\mathcal{N}(c_i),v\neq j}(1-\rho_{old}(v))], j \in \mathcal{N}(c_i)$. We summarize RCSS in Algorithm 1. The output of Algorithm 1 is a suitable rearranged transmission order $\boldsymbol{\pi}$ of output symbols that substantially improves ISRR.

**Algorithm 1** RCSS: proposed output symbol sorting algorithm

Initialize: $\boldsymbol{\pi} = [\,]$, $\boldsymbol{\rho} = [1]_{1\times k}$
**for** counter = 1 to $m$ **do**
  **for** $j = 1$ to $m, j \notin \boldsymbol{\pi}$ **do**
    $p_{dec}(j) = (1-\varepsilon)\sum_{l\in\mathcal{N}(c_i)}[\rho(l)\prod_{v\in\mathcal{N}(c_i),v\neq l}(1-\rho(v))]$
  **end for**
  $i^* = \underset{i}{\operatorname{argmax}}(p_{dec}(i))$
  $\boldsymbol{\pi} = [i^*, \boldsymbol{\pi}]$
  **for** $j \in \mathcal{N}(c_{i^*})$ **do**
    $\rho(j) = \rho_{old}(j)[1-(1-\varepsilon)\prod_{v\in\mathcal{N}(c_{i^*}),v\neq j}(1-\rho_{old}(v))]$
  **end for**
**end for**

Suppose two (or more) output symbols $c_j$ and $c_l$ have equal probability of decoding of an input symbol, i.e., $p_{dec}(j) = p_{dec}(l)$. In addition, assume this probability is the largest probability of decoding an input symbol compared to that of other remaining output symbols. In this case, $\underset{i}{\operatorname{argmax}}(p_{dec}(i))$ returns the index of $c_j$ or $c_l$ whichever has a *lower degree*.

### B. RCSS Lower and Upper Performance Bounds

We investigate the upper and the lower bounds on the performance of RCSS in the following lemmas.

*Lemma 1:* The performance of RCSS is upper bounded by $z = \gamma$ for $\varepsilon \to 0$.

*Proof:* Clearly, we have

$$\lim_{\varepsilon\to 0} p_{dec}(i) = \lim_{\varepsilon\to 0}(1-\varepsilon)\sum_{l\in\mathcal{N}(c_i)}[\rho(l)\prod_{v\in\mathcal{N}(c_i),v\neq l}(1-\rho(v))] \in \{0,1\}, \forall i. \quad (3)$$

This means that since each packet is delivered with high probability the recovery of input symbols is no longer *probabilistic*. Therefore, we have

$$\lim_{\varepsilon\to 0}\rho(j) = \lim_{\varepsilon\to 0}\rho_{old}(j)[1-(1-\varepsilon)\prod_{v\in\mathcal{N}(c_i),v\neq j}(1-\rho_{old}(v))] \in \{0,1\}, \forall j, \quad (4)$$

showing that the recovery of each input symbol is similarly deterministic and is exactly known to the encoder. Therefore, the encoder can determine which output symbols can decode an input symbol with probability 1 in the next step. Consequently, as long as output symbol with $p_{dec}(.) = 1$ are available $z = \gamma$ is obtained. However, since the codes that we designed in Section III may not be capacity-achieving $z = \gamma$ is not necessarily realized. Therefore, the performance of RCSS is indeed upper bounded by $z = \gamma$. ■

We note that if the employed distribution is capacity achieving, i.e., $\gamma_{succ} = 1$, $z = \gamma$ can be obtained.

*Lemma 2:* The performance of RCSS is lower bounded by the performance of [6] (where symbols are only sorted based on their degree) for $\varepsilon \to 1$.

*Proof:* We have $\lim_{\varepsilon\to 1} p_{dec}(i) = 0, \forall i$. Further, since initially we set $\boldsymbol{\rho} = [1]_{1\times k}$ then $\lim_{\varepsilon\to 1}\rho(j) = 1, \forall j$. In other words, $S$ cannot make a meaningful estimate about the recovery of input symbols at $D$. Since for $p_{dec}(i) = p, \forall i$, $\underset{i}{\operatorname{argmax}}(p_{dec}(i))$ returns the index of output symbols with the lowest degree, for $\varepsilon \to 1$ Algorithm 1 boils down to an algorithm that only sorts output symbols based on their degree similar to [6]. Therefore, the performance of RCSS is lower bounded by the performance of the scheme proposed in [6]. ■

### C. Complexity and Delay Incurred by RCSS

It is worth noting that in RCSS all output symbols need to be generated and sorted before the transmission can start in contrast to the *conventional* rateless coding where each $c_i$ can be independently transmitted upon generation. This would result in some delays in transmission when RCSS is employed. However, this delay can be easily eliminated with the following procedure.

Clearly, the order of sorted output symbols is independent of the contents of input symbols and only depends on $\mathcal{N}(c_i), i \in \{1,2,\ldots,m\}$. Therefore, before the transmission starts, $S$ generates $c_i$'s from a *dummy* **x** and obtains and saves an *off-line* version of $\boldsymbol{\pi}_{\text{off-line}}$ and $\mathcal{N}_{\text{off-line}}(c_i)$. When the actual encoding starts, **x** of interest replaces the dummy **x**, and $S$ generates $c_i$, $i \in \boldsymbol{\pi}_{\text{off-line}}$ by XORing $x_j, j \in \mathcal{N}_{\text{off-line}}(c_i)$. In this way, each $c_i$ can be transmitted upon generation and no delay occurs. However, we need to note that the described procedure to eliminate the delay increases the memory requirements and necessitates data storage in contrast to conventional setup.

In addition, when RCSS is employed the overall complexity of rateless coding increases from $O(k)$ [2] in conventional rateless coding to $O(k^2)$ since Algorithm 1 has the complexity of $O(m^2) = O(k^2)$.

### D. Performance Evaluation of RCSS

We implement RCSS for the rateless codes we designed for $\mathbf{W} = (0,0,1)$ with $k = 10^2$ with the distribution $\Omega_1(y) = 0.116y + 0.467y^2 + 0.417y^3$ and plot its ISRR along with its upper and lower bounds (for $\varepsilon \to 1$ and $\varepsilon = 0$, respectively) in Fig. 3. Fig. 3 shows that when an estimate of $\varepsilon$ is available at $S$, RCSS can substantially improve the ISRR of the codes designed in the Section III. For instance at $\gamma = 0.5$ for $\varepsilon = 0.1$, we can see that $z$ has increased from $0.1131$ to $0.4003$.

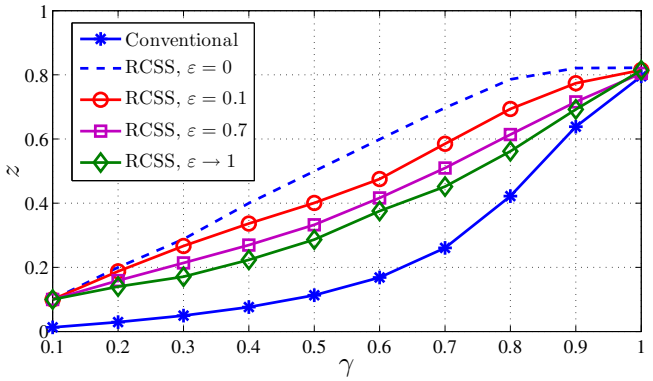


Fig. 3. ISRR of codes designed for $k = 10^2$ with degree distribution $\Omega_1(.)$.

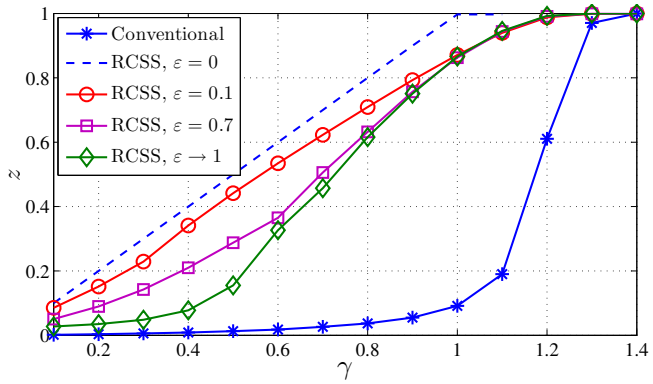


Fig. 4. ISRR of LT codes [1] employing RCSS, and the respective upper and lower bounds.

### E. Employing RCSS With Capacity-Achieving Codes

Since RCSS only reorders the transmission of output symbols, it can be employed along with capacity-achieving rateless codes such as LT codes [1] while preserving their capacity-achieving property. We choose an LT code with parameters $c = 0.05$, $\delta = 0.01$, and $k = 10^3$ ($c$ and $\delta$ are LT codes' distribution parameters [1]) and evaluate its ISRR improvement by RCSS in Fig. 4. Fig. 4 confirms that the ISRR of the employed LT code has considerably improved while its performance at $\gamma_{succ} = 1.4$ has remained intact.

### F. RCSS for Varying $\varepsilon$

Assume that $S$ has generated $m$ output symbols considering $\varepsilon$ and has sorted them employing RCSS. Further, assume that the erasure rate of the channel changes to $\varepsilon_{new}$ when $\frac{k\gamma_c}{1-\varepsilon}$ symbols have already been transmitted and $\frac{k(\gamma_{succ}-\gamma_c)}{1-\varepsilon}$ output symbols are still remaining to be transmitted. If $\varepsilon_{new} > \varepsilon$, less than $k\gamma_{succ}$ output symbols would be collected by $D$, making the full decoding impossible. In this case, $S$ generates $t = (\frac{1}{1-\varepsilon_{new}} - \frac{1}{1-\varepsilon})k(\gamma_{succ} - \gamma_c)$ *new* output symbols and adds them to the queue of output symbols to be transmitted to ensure the delivery of $k\gamma_{succ}$ output symbols to $D$. Next, $S$ *rearranges* all output symbols employing RCSS and continues the transmission. On the other hand, if $\varepsilon_{new} < \varepsilon$ then $S$ randomly drops $1 - \frac{1-\varepsilon}{1-\varepsilon_{new}}$ fraction of remaining output symbols from the transmission queue. Further, if $\varepsilon$ varies multiple times the same procedures are followed after each change.

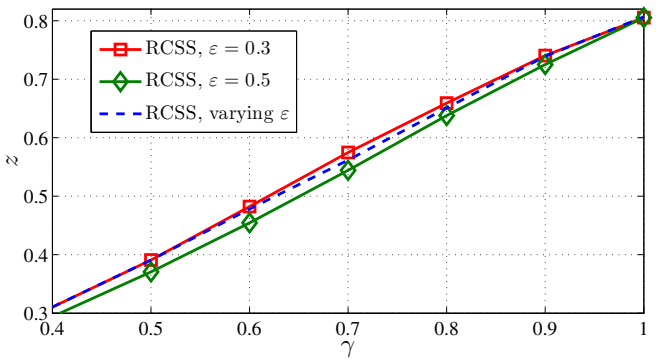


Fig. 5. The resulting ISRR employing RCSS for the case where $\varepsilon$ increases from 0.3 to 0.5 at $\gamma_c = 0.5$.

Assume that $S$ has generated $m$ output symbols employing distribution $\Omega_1(.)$ given in Section IV-D for $\gamma_{succ} = 1$, $\varepsilon = 0.3$, and $k = 10^2$. Further, assume that $\varepsilon$ increases to $\varepsilon_{new} = 0.5$ at $\gamma_c = 0.5$. Therefore, $S$ adds $t = \lceil 0.5714k(\gamma_{succ} - \gamma_c) \rceil$ new output symbols and runs RCSS again. The ISRR of this code has been shown in Fig. 5 where the jump in $\varepsilon_{new}$ occurs at $\gamma = \gamma_c = 0.5$. Fig. 5 shows that a large jump of $66.6\%$ in the $\varepsilon$ is well compensated by RCSS and the same $z$ is achieved at $\gamma_{succ} = 1$. However, due to disturbance in the ordering caused by the newly added symbols a slight performance loss is observed.

## V. Conclusion

Previously, it has been shown that the *intermediate* range of rateless codes is comprised of *three* regions and for each region a rateless coding *distribution* that achieves optimal *intermediate symbol recovery rate* (ISRR) has been designed. In this paper, we selected a point from each region and designed degree distributions that have optimal performance at *all* three selected points employing *multi-objective genetic algorithms*. Next, we assumed that an estimate of the channel erasure rate $\varepsilon$ is available at encoder and proposed *RCSS* that exploits $\varepsilon$ and rearranges the transmission order of output symbols to *further* improve the ISRR of rateless codes. To extend RCSS in our future work, we will consider the probability that output symbols are buffered upon reception and incorporate their effect on rearranging output symbols to be transmitted.

## References

[1] M. Luby, "LT codes," in *Proc. 2002 IEEE Symp. Foundations Computer Science*, pp. 271–280.
[2] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, June 2006.
[3] P. Maymounkov, "Online codes," NYU Technical Report TR2003-883, 2002.
[4] S. Sanghavi, "Intermediate performance of rateless codes," in *Proc. 2007 IEEE Inf. Theory Workshop*, pp. 478–482.
[5] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein, "Growth codes: maximizing sensor network data persistence," in *Proc. 2006 Conf. Applications, Technologies, Architectures, Protocols Computer Commun.*, vol. 36, no. 4, pp. 255–266.
[6] S. Kim and S. Lee, "Improved intermediate performance of rateless codes," in *Proc. 2009 Int. Conf. Advanced Commun. Technol., ICACT*, vol. 3, pp. 1682–1686.

[7] A. Beimel, S. Dolev, and N. Singer, "RT oblivious erasure correcting," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1321–1332, 2007.
[8] A. Talari and N. Rahnavard, "Rateless codes with optimum intermediate performance," in *Proc. 2009 IEEE Global Telecommun. Conf.*, pp. 1–6.
[9] A. Talari, B. Shahrasbi, and N. Rahnavard, "Efficient symbol sorting for high intermediate recovery rate of LT codes," in *Proc. 2010 IEEE Int. Symp. Inf. Theory*, pp. 2443–2447.
[10] M. G. Luby, M. Mitzenmacher, and M. A. Shokrollahi, "Analysis of random processes via and-or tree evaluation," in *Proc. 1998 ACM-SIAM Symp. Discrete Algorithms*, pp. 364–373.
[11] N. Rahnavard, B. Vellambi, and F. Fekri, "Rateless codes with unequal error protection property," *IEEE Trans. Inf. Theory*, vol. 53, no. 4, pp. 1521–1532, Apr. 2007.
[12] N. Rahnavard and F. Fekri, "Generalization of rateless codes for unequal error protection and recovery time: Asymptotic analysis," in *Proc. 2006 IEEE Int. Symp. Inf. Theory*, pp. 523–527.
[13] R. Karp, M. Luby, and A. Shokrollahi, "Finite length analysis of LT codes," in *Proc. 2004 Int. Symp. Inf. Theory*, p. 39.
[14] E. Maneva and A. Shokrollahi, "New model for rigorous analysis of LT-codes," in *Proc. 2006 Int. Symp. Inf. Theory*, pp. 2677–2679.
[15] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[16] Available: http://cwnlab.ece.okstate.edu/
[17] R. Gummadi and R. Sreenivas, "Relaying a fountain code across multiple nodes," in *Proc. 2008 IEEE Inf. Theory Workshop*, pp. 149–153.

**Ali Talari** received his B.S. degree in electrical engineering from Kashan University, Kashan, Iran, in 2003 and his M.S. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 2006. Ali joined the School of Electrical and Computer Engineering at Oklahoma State University as a Ph.D. student in January 2008. His research interests are novel error control coding techniques, communications theory, signal processing in wireless sensor networks, and compressive sensing techniques.

**Nazanin Rahnavard** (S'97-M'10) received her B.S. and M.S. degrees in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 1999 and 2001, respectively. She then joined the Georgia Institute of Technology, Atlanta, GA, in 2002, where she received her Ph.D. degree in the School of Electrical and Computer Engineering in 2007. Dr. Rahnavard joined the School of Electrical and Computer Engineering at Oklahoma State University as an Assistant Professor in August 2007. She has interest and expertise in a variety of research topics in the communications and networking areas. She is particularly interested in modern error-control coding techniques and their applications, compressive sensing, cognitive radio networks, and ad-hoc/sensor networks.

Dr. Rahnavard received an NSF CAREER Award in 2011. She is also the recipient of the 2007 Outstanding Research Award from the Center of Signal and Image Processing at Georgia Tech. She serves on the editorial boards of the *Elsevier Journal on Computer Networks* (COMNET) and on the Technical Program Committee of several prestigious international conferences.