Two Degree Forest Based LT Codes with Feedback

Liang Liu[1,2,3,4(✉)] and Feng Liu[1,2,3,4]

[1] School of Electronic and Engineering, Beihang University, Beijing, China
{Liuliangl945, liuf}@buaa.edu.cn
[2] The Collaborative Innovation Center of Geospatial Technology, Wuhan, China
[3] Beijing Key Laboratory for Network-Based Cooperative Air Traffic Management (No. BZ0272), Beijing, China
[4] Beijing Laboratory for General Aviation Technology, Beijing, People's Republic of China

Abstract. The performance of belief propagation (BP) decoding algorithm for LT codes is significantly deteriorated, as the data-block length decreases, since the randomly encoding of LT codes causes lots of wasted output symbols, which is helpless for decoding. To solve this problem, this paper provides two degree forest based LT codes in order to help the sender to send badly needed symbols to accelerate decoding throughout entire receiving process. Through gathering two degree output symbols into separable trees, the decoder can easily get and feedback the indexes of the badly needed input symbols. Simulation results show that, in the short data-block length case, two degree forest based LT codes achieve much lower coding overhead, consume much smaller storage resources, and need less feedback opportunities compared with current LT codes algorithms.

Keywords: LT codes · Two degree forest · Feedback channel · Robust soliton distribution

1 Introduction

LT codes are the simplest kind of fountain codes [1] with capacity achieving performance on erasure channels [1–3]. The biggest advantage of LT codes is that it is simultaneously near optimal for different erasure channels with time varying or unknown erasure rate [1]. Moreover, due to the character of sparsity for encoding [1], the decoder can achieve linear decoding complex, according to a belief propagation (BP) decoding algorithm. In addition, only one single bit feedback is needed from the decoder to the sender during entire transmission process. Due to above advantages, LT codes are widely used in different scenarios of communications.

However, it should not be ignored that the performance of LT codes is compromised considerably when the data block length is relatively small [1, 4, 5]. Such situation happens frequently. For one thing, in some applications, it is not needed to transmit large amount of data bits. For another, in some energy limited equips, such as

Q. Chen et al. (Eds.): ChinaCom 2016, Part I, LNICST 209, pp. 232–241, 2018.
DOI: 10.1007/978-3-319-66625-9_23

small wireless sensors [9] and satellites in low earth orbits [10], there are not sufficient storage resource to buffer large amount of data bits. Hence, it is needed to enhance the performance of LT code when data block length is relatively small.

To enhance the performance of LT code with relatively small data block length, former researchers have proposed some improvements based on limited feedback. In the work by [6], the decoder transmits feedback message about the number of original symbols that are already recovered, so that the sender can shift the degree distribution of LT code to accelerate data recovery, such method is called shifted LT code or SLT code for short. However, the performance of SLT code is limited due to following two aspects. For one thing, although average degree increases when some original symbols are recovered at decoder, it still has large probability that newly generated output symbols are useless for decoder. For another, as average degree increases, the probability of degree one decreases, which is negative for traditional BP decoding algorithm.

To better accelerate the decoding process, LT codes with alternating feedbacks (LT-AF codes) is proposed [7, 8]. In this algorithm, not only the currently decoded number of original symbols, but also the needed input symbol index is feedback from the decoder, to further enhance decoding process. The second kind of feedback is useful, if some input symbol is badly needed for the decoder. To better figure out the suitable input symbol index, three different principles are proposed is [8]. However, in [7, 8], the feedback message about index of input symbol is only used when the number of output symbols received by decoder is larger than the number of original symbols, which is negative for decoding process. Since traditional decoding algorithm can only recover a small fraction of input symbols, when number of received output symbols is less than the number of input symbols [1], then it is inevitable that the storage costs for decoder is huge. Moreover, it is reasonable for the decoder to obtain the badly needed input symbols in the whole receiving and decoding process.

This paper proposes the two degree forest based LT codes (TBLT codes) with feedback to enhance the performance of LT codes with relatively small data block length. TBLE codes work as follows. First, in order to provide appropriate feedback for indexes of input symbols, the two degree forest (TDF), which consists of output symbols with two adjacent input symbols, is formulated and maintained throughout entire decoding process. The feedback message about the badly needed index of input symbol is sent to the sender, as long as the size of some two degree tree (TDT) is larger than certain threshold. If the badly needed input symbol arrives at destination, all input symbols, which is included in that TDT, can be recovered, and the iterative decoding algorithm can be accelerated to recovery more input symbols. Notice that, although the idea of two degree chains is mentioned in [8] as one kind of principle to get the indexes of needed input symbols, which is similar to the two degree trees in our methods, the function of two degree forest is not fully explored and analyzed in [8].

Simulation results show that TBLT codes have following advantages. First, the average coding overhead, which is defined as proportion of additional data bits needed for transmission, can be reduced by about 40%, compared with traditional LT codes and SLT codes [6]. Second, the peak value of restore cost during entire decoding

process can be reduced by about 35%. Finally, compared with SLT codes, TBLT codes require much less feedback times.

The paper is organized as follows. Section 2 presents basic knowledge about LT codes. Section 3 presents the design of two degree forest based LT codes. The simulation results with comparison are given in Sect. 4. Section 5 concludes the paper.

2 Preliminary

First, for simplicity without losing generalization, assume that all input and output symbols are binary symbols, which may include thousands of bits. The encoding and decoding procedures of LT codes are as follows.

LT Encoding: Let k be the number of input symbols for encoding. Let $U = \{u_1, \ldots, u_k\}$ be the set of input symbols. First, an output symbol degree d is chosen from the Robust-Soliton distribution [1], $M = \{\mu_1, \mu_2, \ldots, \mu_k\}$, where μ_i is the probability that $d = i$, such that $\sum_{i=1}^{k} \mu_i = 1$. Next, uniformly and randomly chose d input symbols. Then the output symbol v is generated through XORing those d input symbols. Those d input symbols refer to the neighbors of v. The encoding procedure repeats until the single-bit feedback comes. The RS distribution is obtained by combining ideal-soliton (IS) distribution ρ_i and distribution τ given by

$$\rho_i = \begin{cases} 1/k & i = 1 \\ 1/\mathrm{i}(\mathrm{i}-1) & i = 2, \ldots, k \end{cases} \tag{1}$$

and

$$\tau_i = \begin{cases} R/ik & i = 1, \ldots, \mathrm{k}/\mathrm{R} - 1 \\ \mathrm{R}/\mathrm{k}\ln(\mathrm{R}/\delta) & i = k/R \\ 0 & i = k/R + 1, \ldots, k \end{cases}$$

respectively, where $R = c\ln(k/\delta)\sqrt{k}$, δ and c are two tunable parameters. Then, RS distribution is obtained through

$$\mu_i = \frac{\rho_i + \tau_i}{\beta}, i = 1, \ldots, k, \tag{2}$$

where, $\beta = \sum_{i=1}^{k} \rho_i + \tau_i$.

LT Decoding: Currently, the basic decoding algorithm for LT codes is the belief propagation, which works iteratively as follows. When decoder finds an output symbol, such that the value of all but one of its neighboring input is known, the unknown input symbol can be obtained by bitwise XOR operations and removes all edges incident to that output symbol. This process is repeated until no such output symbols exist. Notice

that the set of output symbols that are reduced to degree one is called ripple, which is critical for LT decoding. If the ripple is empty, the decoding stops and waits for new output symbols of degree one to proceed the decoding. If all k input symbols are recovered, the decoder sends a single-bit feedback message to the sender to inform the success of decoding.

Let γ be the number of output symbols during transmission. Let γ_S be the average number of output symbols when successful recovery of k input symbols can be obtained. As shown in [1], $\gamma_S \geq k$ holds. Let η and ε be the redundancy and overhead for LT codes, such that $\eta = \gamma_S/k$, and $\varepsilon = \eta - 1$. It is obvious that $\eta \geq 1$ and $\varepsilon \geq 0$.

Although LT codes with RS distribution can asymptotically achieve, i.e. $\eta \to 1$, as $k \to \infty$, η is significantly larger than 1 when k is finite [4, 5]. Hence, it is needed to improve the performance when k is finite.

3 Design of Two Degree Forest Based LT Codes

In this section, the two degree forest based LT (TBLT) codes are proposed. The encoding process is the same as the LT codes, while the decoding process is changed. First, the structure of two degree forest for decoder is introduced. Then, the updating procedure for two degree forest is proposed. Finally, based on the feedback rule, the decoding algorithm for TBLT is proposed.

3.1 Two Degree Forest

As mentioned before, the decode process can be promoted considerably if appropriate input symbol can be sent to destination. However, which input symbol is most important for decoder is unclear. To solve this problem, the two degree forest of output symbols is formulated at decoder as follows.

Definition 1. *Two degree tree (TDT)*: a TDT, Γ, is a tree, which has following properties: First, the nodes of the tree are output symbols, which have exact two neighbors as input symbols. Second, let v_1 and v_2 be two nodes in Γ, then one link exists between v_1 and v_2, if and only if v_1 and v_2 have exactly one same neighbor. Third, as a tree, Γ has no loop path.

Moreover, let V_Γ and U_Γ be the set of output symbols in Γ and set of neighbor input symbols in Γ, respectively. Let $|V_\Gamma|$ and $|U_\Gamma|$ be the number of elements in V_Γ and U_Γ respectively, then it is obviously that $|U_\Gamma| = |V_\Gamma| + 1$.

Definition 2. *Two degree forest (TDF)*: a TDF, Ω, is a graph consists of two degree trees, such that $\Omega = \{\Gamma_1, \ldots \Gamma_{N_\Omega}\}$, which has following property. Let $\Gamma^{(1)}$ and $\Gamma^{(2)}$ be any two TDTs in Ω. Let $v^{(1)} \in V^{(1)}$ and $v^{(2)} \in V^{(2)}$ be any two output variables, which lie in $\Gamma^{(1)}$and $\Gamma^{(2)}$ respectively. Then, $v^{(1)}$ and $v^{(2)}$ have no same neighbor input symbol.

For example as shown in Fig. 1, a TDF with two TDTs is formulated at decoder.

The most important feature for TDF is that, for each TDT, $\Gamma \in \Omega$, the input symbols in U_Γ can be recovered with breadth first search, if any input symbol in U_Γ is obtained. For example as shown in Fig. 2, if the input symbol, u_2 is obtained, then u_4 can be recovered, since u_2 and u_4 are neighbor input symbols of the output symbol indexed by 2,4 in Fig. 2. Next, u_{14}, u_8 and u_{12} can be recovered, since output symbols indexed by 2,14, 2,8, and 4,12 are adjacent to the output symbol indexed by 2,4. Finally, u_{18} and u_{16} can be recovered, since output symbols indexed by 8,18 is adjacent to output symbols indexed by 2,8, and output symbols indexed by 4,16 is adjacent to output symbols indexed by 4,12.

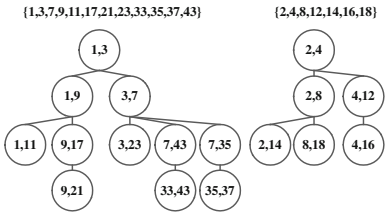

Fig. 1. Simple example of two degree forest.

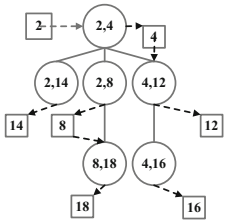

Fig. 2. Recovery of input symbols within one two degree tree

3.2 Updating Two Degree Forest

In the receiving and decoding process, the two degree forest is updated according to following operations.

(1) Tree adding: Let $\Gamma^{(1)} \in \Omega$. Let $\widetilde{v}$ be the new output symbol that reaches destination, such that $\widetilde{v}$ has two neighbor input symbols, $\tilde{u}_1$ and $\tilde{u}_2$. If $\tilde{u}_1 \in U_{\Gamma^{(1)}}$, and $\tilde{u}_2 \notin U_{\Gamma^{(1)}}$, then $\Gamma^{(1)}$ can be added with a new output symbol such that $V_{\Gamma^{(1)}} = V_{\Gamma^{(1)}} \cup \{\widetilde{v}\}$ and $U_{\Gamma^{(2)}} = U_{\Gamma^{(2)}} \cup \{\tilde{u}_2\}$.

(2) Tree combining: Let $\Gamma^{(1)}$, $\Gamma^{(2)} \in \Omega$. Let $\widetilde{v}$ be the new output symbol that reaches destination, such that $\widetilde{v}$ has two neighbor input symbols, $\tilde{u}_1$ and $\tilde{u}_2$. If $\tilde{u}_1 \in U_{\Gamma^{(1)}}$ and $\tilde{u}_2 \in U_{\Gamma^{(2)}}$, then $\Gamma^{(1)}$ and $\Gamma^{(2)}$ can be combined to formulate a new tree $\tilde{\Gamma}$, such that $V_{\tilde{\Gamma}} = V_{\Gamma^{(1)}} \cup V_{\Gamma^{(2)}} \cup \{\widetilde{v}\}$, and $U_{\tilde{\Gamma}} = U_{\Gamma^{(1)}} \cup U_{\Gamma^{(2)}}$.

(3) Tree removing: Let $\Gamma^{(1)} \in \Omega$ be one TDT, such that all input symbols in $U_{\Gamma^{(1)}}$ are recovered by decoder, then $\Gamma^{(1)}$ is removed from TDF, namely, $\Omega = \Omega/\Gamma^{(1)}$.

Entire updating procedure for two degree forest is as follows.

Algorithm 1: TDF Updating Procedure

Input: Current TDF $\Omega = \{\Gamma_1, ..., \Gamma_N\}$, one output symbol $\tilde{v}$ with two neighbor input symbols, $\tilde{u}_1$ and $\tilde{u}_2$.

If two TDT $\Gamma^{(1)}, \Gamma^{(2)} \in \Omega$, such that $\tilde{u}_1 \in U_{\Gamma^{(1)}}$, and $\tilde{u}_2 \notin U_{\Gamma^{(1)}}$:

Then, get one combined tree, $\tilde{\Gamma}$, according to tree combining operation. Let $\Omega^* = \Omega \bigcup \tilde{\Gamma}$, and let $\tilde{\Omega} = \Omega^* / \Gamma^{(1)} \bigcup \Gamma^{(2)}$.

Else if one $\Gamma^{(1)} \in \Omega$ exists, such that $\tilde{u}_1, \tilde{u}_2 \in U_{\Gamma^{(1)}}$:

Then, discard $\tilde{v}$, and let $\tilde{\Omega} = \Omega$.

Else if only one $\Gamma^{(1)} \in \Omega$ exists, such that $\tilde{u}_1 \in U_{\Gamma^{(1)}}$, and $\tilde{u}_2 \notin U_{\Gamma^{(1)}}$:

Then, add $\Gamma^{(1)}$ with $\tilde{v}$, according to tree adding operation.

Else if no $\Gamma^{(1)} \in \Omega$ exists, such that $\tilde{u}_1 \in U_{\Gamma^{(1)}}$, or $\tilde{u}_2 \in U_{\Gamma^{(1)}}$:

Then, formulate one new TDT, Γ^*, such that $V_{\Gamma^*} = \{\tilde{v}\}$ and $U_{\Gamma^*} = \{\tilde{u}_1, \tilde{u}_2\}$.

Output: $\tilde{\Omega}$

3.3 Feedback Rule and Decode Process

As mentioned before, for any $\Gamma^{(1)} \in \Omega$, all the input symbols within $U_{\Gamma^{(1)}}$ can be recovered with a breadth first search, if one of them is obtained. Furthermore, more input symbols can be recovered iteratively with those input symbols within $U_{\Gamma^{(1)}}$, according to the belief propagation decoding algorithm. Hence, it is reasonable for the decoder to inform the sender to send one symbol $\tilde{u} \in U_{\Gamma^{(1)}}$, when $|U_{\Gamma^{(1)}}|$ is large enough. The feedback rule is as follows.

Let $\beta > 0$ be one integer. Then, if one $\Gamma^{(1)} \in \Omega$ exists, such that $|U_{\Gamma^{(1)}}| \geq \beta$, the decoder send a feedback message to the sender, that the sender should send one original input symbol $\tilde{u}$, such that $\tilde{u} \in U_{\Gamma^{(1)}}$. Such feedback rule is called $F_\beta(\tilde{u})$, where β is called the feedback threshold.

Based on the two degree forest and feedback rule, the decoding algorithm is proposed as follows.

Algorithm 2: TDF Based Decoding Algorithm for TBLT Codes

Input: Current TDF $\Omega = \{\Gamma_1, ..., \Gamma_N\}$, current set of output symbols, V, and current set of recovered symbols $\widetilde{U} \subset U$, newly arrived output symbol $\tilde{v}$, with the set of index of neighbor input symbols $\Lambda_{\tilde{v}}$, and β.

If $\left|\Lambda_{\tilde{v}}\right| \geq 3$: let $V = V \bigcup \{\tilde{v}\}$.

Else if $\left|\Lambda_{\tilde{v}}\right| = 2$:

Then update Ω according to algorithm1.

If one $\Gamma^{(1)} \in \Omega$ exists, such that $\left|U_{\Gamma^{(1)}}\right| \geq \beta$: send $F_\beta(\tilde{u})$ to sender.

Else if $\left|\Lambda_{\tilde{v}}\right| = 1$:

Take BP decoding algorithm to obtain new $\widetilde{U}$ and V.

If $\widetilde{U} = U$:

Then, inform the sender that decoding succeeds.

Else if $\left|\widetilde{U}\right| < |U|$:

Then, for any $\Gamma \in \Omega$, if $U_{\Gamma^{(1)}} \subset \widetilde{U}$, let $\Omega = \Omega / \{\Gamma\}$.

Compared with the traditional BP decoding algorithm, additional cost for TDF based decoding algorithm is small. For one thing, the two degree forest doesn't need much storage resource, since, the decoder only needs to restore the index information of input symbols and output symbols to maintain the structure for TDF. For another, since each output symbol in TDF has exactly two adjacent input symbols, it doesn't need much calculation cost to put each output symbols into suitable TDT.

4 Simulation Results and Analysis

As mentioned before, the basic aim for the proposed algorithm in this paper is to reduce the redundancy of LT codes, when k is relatively small, hence, let k = 500, 1000, in simulation. Next, the classical robust soliton distribution is used through simulation, where it is set that δ = 0.1, and c = 0, 0.1, 0.2 for comparison. Moreover, in our algorithm, the feedback threshold, β, is set to be 10, 20, 30, 40, 50 for comparison.

Following two algorithms are used for comparison:

(1) The classical BP decoding algorithm for LT codes [1].
(2) The SLT codes [6], which is based on the idea that relative larger degree should be generated, if the sender is informed of the number of input symbols that have already recovered.

The simulation results are obtained through getting average values of 10000 tests, for each simulation condition.

Comparison of simulation results for overhead is proposed in Fig. 3. TBLT codes achieve lower overhead than traditional LT codes and SLT codes, no matter what feedback threshold value is. When k = 500, compared with the best case of SLT codes that the overhead is about 0.25, the best case of TBLT codes is about 0.15, which is about 40% lower. Moreover, it should be noticed that TBLT codes get the smallest overhead, when c = 0, while LT codes and SLT codes get the smallest overhead when c = 0.1. Such results can be explained as follows. The two degree output symbols are of great importance for the decode process in TBLT codes. However, according to the definition of robust soliton distribution [1], the probability of degree two decreases as c increases, which is negative for the efficiency of two degree forest.

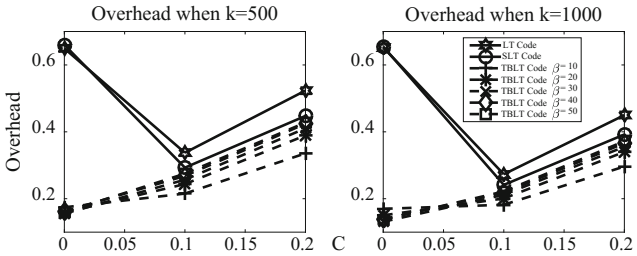

Fig. 3. Comparison of simulation results of overhead when packet error rate is zero of degree

Comparison of simulation results of overhead when packet error rate is larger than 0 is shown in Fig. 4. It is naturally that, the larger packet error rate is, the larger the overhead is. The overhead of TBLT codes are still lower than that of the traditional LT codes and SLT codes under different c values, no matter what the packet error rate is.

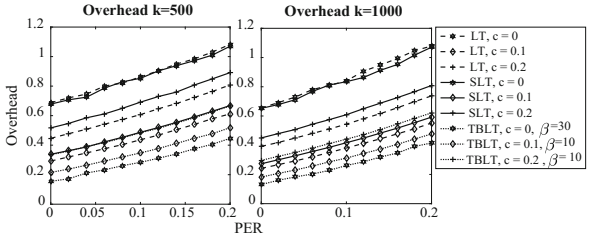

Fig. 4. Comparison of simulation results of overhead when packet error rate is larger than 0.

The simulation results for decode process is expressed through the relationship between average decoded number of input symbols and the number of output symbols received at decoder, as shown in Fig. 5. For simplicity, the best case, c = 0.1, is used for LT codes and SLT codes. While, when c = 0, TBLT codes achieve smallest overhead. Compared with LT codes and SLT codes, TBLT codes can accelerate the decoding process. When number of received output symbols at decoder equals k, LT and SLT can only recover a small fraction of input symbols, while, TBLT can recover majority of input symbols.

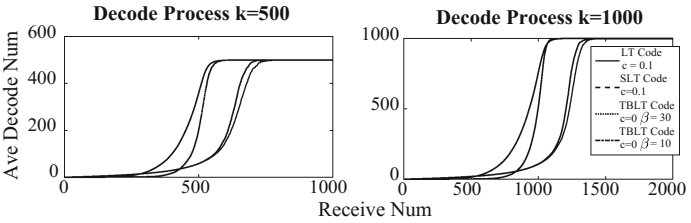

Fig. 5. Comparison of simulation results for the relationship between average decoded number of input symbols and the receive number of output symbols.

Simulation results for the relationship between average stored output symbols and the number of received output symbols at decoder is given by Fig. 6. Compared with LT codes and SLT codes, the storage resource can be saved considerably by TBLT algorithm. For one thing, compared with LT codes and SLT codes, the peak value of storage resource is reduced by about 35% for TBLT codes. For another, the amount of storage resource of TBLT codes reduce much early than that of LT codes and SLT codes, which increases the efficiency of buffer's usage.

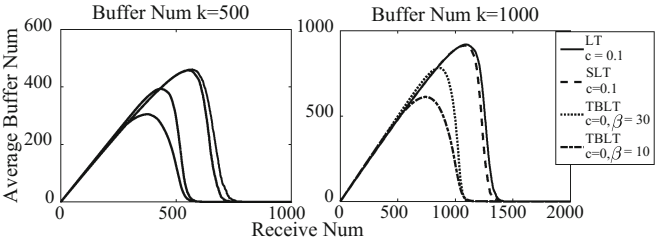

Fig. 6. Comparison of simulation results for the relationship between average stored output symbols and the number of received output symbols at decoder.

Simulation results for the relationship between average feedback times and the number of received output symbols at decoder is given by Fig. 7. Compared with SLT

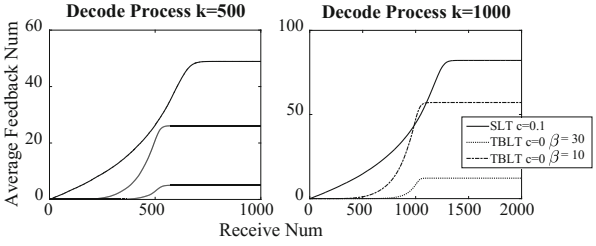

Fig. 7. Comparison of simulation results for the relationship between average feedback times and the number of received output symbols at decoder.

codes, TBLT codes require much less feedback times. Especially, when $\beta = 30$, the average feedback times can be reduced by about 80%, compared with SLT codes.

5 Conclusion

This paper proposes two degree forest based LT codes to enhance the performance of LT codes in the relatively small data-block length case. To help the sender to send the appropriate output symbols to accelerate decoding, the two degree output symbols are gathered into separable trees, so that decoder can get the indexes of badly needed input symbols if the size of some tree is larger than certain threshold value.

Simulation results show that TBLT codes can reduce coding overhead by about 40%, compared with traditional LT codes and SLT codes. Moreover, the storage cost during entire decoding process can be reduced considerably. Finally, compared with SLT codes, TBLT codes require much less feedback opportunities.

Acknowledgments. This work is supported in part by National Natural Science Foundation of China (Grant Nos. 61231013, 91438206, 91538202 and 61521091) and Fundamental Research Funds for the Central Universities (Grant No. YMF-14-DZXY-027).

References

1. Luby, M.: LT codes. In: The 43rd Annual IEEE Symposium on Foundations of Computer Science, pp. 271–280 (2002)
2. Shokrollahi, A.: Raptor codes. IEEE Trans. Inf. Theory **52**, 2551–2567 (2006)
3. Maymounkov, P.: Online codes, NYU Technical report TR2003-883 (2002)
4. Bodine, E., Cheng, M.: Characterization of Luby transform codes with small message size for low-latency decoding. In: IEEE International Conference on Communications, ICC, pp. 1195–1199 (2008)
5. Hyytia, E., Tirronen, T., Virtamo, J.: Optimal degree distribution for LT codes with small message length. In: 26th IEEE International Conference on Computer Communications, pp. 2576–2580 (2007)
6. Hagedorn, A., Agarwal, S., Starobinski, D., Trachtenberg, A.: Rateless coding with feedback. In: INFOCOM, pp. 1791–1799 (2009)
7. Talari, A., Rahnavard, N.: LT-AF codes: LT codes with alternating feedback. In: IEEE International Symposium on Information Theory Proceedings, pp. 2646–2650 (2013)
8. Talari, A., Rahnavard, N.: Robust LT codes with alternating feedback. Comput. Commun. **49**(1), 60–68 (2014)
9. Younis, O., Fahmy, S.: HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. IEEE Trans. Mob. Comput. **3**(4), 366–379 (2004)
10. Hu, Y.F., Berioli, M., Pillai, P., Cruickshank, H., Giambene, G., Kotsopoulos, K., Guo, W., Chan, P.M.L.: Broadband satellite multimedia. IET Commun. **4**(13), 1519–1531 (2010)