UNEQUAL LOSS-PROTECTED MULTIPLE DESCRIPTION CODING OF SCALABLE SOURCE STREAMS USING A PROGRESSIVE APPROACH

Majid R. Ardestani[1], Alireza Aminlou[2] and Ali Asghar Beheshti Shirazi[1]

[1]Electrical Engineering Department, Iran University of Science and Technology (IUST), Tehran, Iran,
{m_ardestani, abeheshti}@iust.ac.ir
[2]Electrical and Computer Engineering Department, University of Tehran (UT), Tehran, Iran,
a.aminlou@ece.ut.ac.ir

ABSTRACT

An analysis-based approach for unequal loss-protected multiple description coding (packetization) of the scalable (prioritized / progressive) source code streams is proposed. For a given number of packets (descriptions) of the known size, unequal loss-protected packetization leads to segment the scalable code stream, such that the source can be reconstructed with the maximum possible fidelity at the decoder side. Here, we find an analytical relation between optimal sizes of any two consecutive segments. This idea yields a low-complexity progressive solution with a performance close to that of local search [1], which has been approved as an efficient method to solve the segmentation problem. Simulation results are used to confirm the efficiency of the proposed method as compared with the local search algorithm.

Index Terms— Unequal loss-protected packetization, multiple description coding, scalable code stream, joint source-channel coding

1. INTRODUCTION

Transmitting an embedded source bit stream (e.g. image and video scalable bit stream) through a packet erasure network requires an appropriate packetization scheme so that the different parts of the data stream with different levels of importance are unequally protected against packet loss. Forward error correction based multiple description coding (MD-FEC), proposed by Puri et al. [3], is an efficient packetization scheme with the capability of unequal loss protection (ULP).

ULP is a joint source-channel coding problem in which an embedded source code stream with a known distortion-rate characteristic is adaptively segmented according to the packet loss probability distribution function (PDF) of the channel. Then, each segment is protected with systematic error correction codes (e.g. Reed-Solomon code). Afterwards, a given number N of equally important packets of the fixed length L is generated, each carrying an equal contribution of all the protected segments. The segmentation process of the embedded bit stream should be carried out in such a way that the maximum reconstruction fidelity is obtained at the decoder side.

Many researchers have devised different solutions for the ULP segmentation problem. In [3], Puri and Ramchandran proposed a Lagrange multiplier-based algorithm. In [2], Dumitrescu et al. proposed an $O(N^2L^2)$ algorithm that is close to optimal in general case and optimal if the distortion-rate fidelity function is convex and the packet loss probability function is monotonically decreasing. Mohr et al [4] proposed a suboptimal search algorithm. In [1] Stankovic et al. proposed an $O(NL)$ local search algorithm that starts from a solution maximizing the expected number of received source bits and iteratively improves this solution.

Recently, joint source-channel coding is extended to be used in applications such as three-dimensional (3D) TV and video gaming in which 3D objects should be coded in an efficient and error-resilient manner. There has been a great amount of research for efficient progressive compression of 3D data sources [6]. MD-FEC has been recently applied for error-resilient transmission of 3D data meshes through error-prone networks [7][8][9].

In this work, we analytically derive a relation between optimal sizes of any two consecutive segments. This idea simply enables us to progressively approximate the optimal size of each segment from the previous one. In this way, each valid value for the size of the first (the most important) segment initiates a progressive process. To keep the complexity of the search procedure reasonable, we find a short interval around the optimal value for the size of the first segment through an optimization analysis.

The rest of this paper is organized as follows. In section 2, MD-FEC problem formulation is stated. The proposed progressive method will be discussed in section 3. The simulation results are presented in section 4 to compare the proposed method with the local search algorithm in[1]. Finally, section 5 concludes the paper.

2. MD-FEC PROBLEM FORMULATION

In this section, we state the problem of segmentation of a scalable source code stream (which is previously generated using some scalable source encoder, e.g. JPEG2000 for image coding) for ULP packetization. Our notations are essentially the same as [1,2]. We want to obtain N equally important packets (descriptions) of L symbols (e.g. bytes) each, from the embedded code stream. First, we partition the embedded code stream into L segments with non-decreasing sizes of $m_1 \leq m_2 \leq \ldots \leq m_L$ symbols. Each segment j is protected with $f_j = N - m_j$ parity symbols using (N, m_j) systematic Reed-Solomon code. Then, the i'th symbols of all the protected segments are grouped to form the i'th description, $i = 1, 2\ldots N$. If n descriptions are lost at the decoder side, such that $f_j \geq n > f_{j+1}$, then the first j segments of the source code stream can be reconstructed.

Let p_n stand for the probability of losing exactly n packets (out of N) and $c(k) = \sum_{n=0}^{k} p_n$, $k = 0\ldots N$. Then, $c(f_j)$ is the probability that the decoder correctly reconstructs the source up to j'th segment. Let $d(r)$ denote the distortion-rate function of the scalable source code stream and let X stand for a random variable whose value is the number of lost packets. The optimal MD-FEC problem consists of finding a parity vector $\underline{f} = (f_1, \ldots, f_L)$ that minimizes the expected distortion

$$D_{avg}\left(\underline{f}\right) = \sum_{j=0}^{L} P_j\left(\underline{f}\right) d\left(r_j\right) \quad (1)$$

where $P_0(\underline{f}) = P(X > f_1) = 1 - c(f_1)$, $P_j(\underline{f}) = P(f_{j+1} < X \le f_j) = c(f_j) - c(f_{j+1})$, for $j = 1, .., L$-1, $P_L(\underline{f}) = P(X \le f_L) = c(f_L)$, $r_0 = 0$, $r_j = \sum_{k=1}^{j} m_k = jN - \sum_{k=1}^{j} f_k$, for $j = 1, .., L$.

Although $f_1 \ldots f_L$, $r_1 \ldots r_L$, $c(.)$, and $d(.)$ are discrete-valued quantities, however, we will virtually treat them as continuous-valued quantities, so that a meaningful derivation can be normally applied. In this manner, $c^{'}(.)$ and $d^{'}(.)$ denote the derivatives of $c(.)$ and $d(.)$, respectively, with respect to their input arguments. Note that $c'(f)$ can be roughly estimated by p_f.

3. PROPOSED PROGRESSIVE METHOD

In this section, we will discuss the proposed progressive solution for the optimization problem stated in section 2. To solve this problem in an optimal manner, it is necessary to search among all candidate parity vectors $\underline{f} = (f_1 \ldots f_L)$ with successively non-increasing elements and then select the one that minimizes the expected distortion in (1). This full search method is not applicable for real-time applications, especially for large values of N and L. In this section, we will derive an optimality condition which provides a rather simple relation between any two consecutive elements of $\underline{f}$. This optimality condition yields a low complexity progressive approach.

Let $D_i(x)$, $i = 1 \ldots L$-1, denote the expected distortion of the received source bit stream when two consecutive elements, f_i and f_{i+1}, of the optimal solution, $\underline{f}$, are changed by a minute positive amount of x. That is,

$$D_i\left(x\right) = D_{avg}\left(\left\{\ldots, f_i + x, f_{i+1} - x, \ldots\right\}\right) \quad (2)$$

$D_i(x)$ is defined such that the overall parity budget remains unchanged and only a limited number of parameters of $\underline{f}$ and $\underline{r}$ are affected. In fact, f_i and m_{i+1} (f_{i+1} and m_i) are increased (decreased) by x, and r_i is decreased by x. The other parameters remain unchanged. The optimality condition of $\underline{f}$ imposes

$$I_i\left(\underline{f}\right) \overset{\Delta}{=} \frac{\partial D_i\left(x\right)}{\partial x}\Big|_{x=0} = 0 \quad (3)$$

for $i = 1, .., L$-1. Considering the changing parameters in (3) (i.e., f_i, f_{i+1} and r_i), we have

$$I_i\left(\underline{f}\right) = \frac{\partial\left[P_{i-1}\left(\underline{f}\right)d\left(r_{i-1}\right) + P_i\left(\underline{f}\right)d\left(r_i\right) + P_{i+1}\left(\underline{f}\right)d\left(r_{i+1}\right)\right]}{\partial x}\Big|_{x=0}$$

$$= \frac{\partial\left[-c\left(f_i\right)d\left(r_{i-1}\right) + \left(c\left(f_i\right) - c\left(f_{i+1}\right)\right)d\left(r_i\right) + c\left(f_{i+1}\right)d\left(r_{i+1}\right)\right]}{\partial x}\Big|_{x=0}$$

As mentioned above, f_i (f_{i+1} and r_i) is increased (are decreased) by x. Hence, $(\partial f_i / \partial x) = 1$, and $(\partial f_{i+1} / \partial x) = (\partial r_i / \partial x) = -1$. Finally, the optimality condition is derived as follows

$$\begin{aligned} I_i\left(\underline{f}\right) &= -c^{'}\left(f_i\right)d\left(r_{i-1}\right) + \left[c^{'}\left(f_i\right) + c^{'}\left(f_{i+1}\right)\right]d\left(r_i\right) \\ &\quad - \left[c\left(f_i\right) - c\left(f_{i+1}\right)\right]d^{'}\left(r_i\right) - c^{'}\left(f_{i+1}\right)d\left(r_{i+1}\right) \\ &= 0 \end{aligned} \quad (4)$$

for $i = 1, .., L$-1. The equation (4) is dependent to only a limited number of parameters including f_i, f_{i+1}, r_{i-1}, r_i and r_{i+1}. However, the parameters r_k are calculated based on the f_j values, j=1, .., k. We noted that the value of f_{i+1} can be calculated by satisfying (4), when all the previous values of f_j, $j = 1 \ldots i$, are available. Hence, we propose to calculate f_i values starting from $i = 1$ toward $i = L$, step-by-step and "progressively". In fact, the equation (4) provides a relation between any two consecutive elements of the "optimal" parity vector. It means, if we have f_1, then we can find f_2. When f_2 is found, f_3 will be subsequently determined, and so on. Therefore, if we have the first element of the optimal solution, we can progressively find the others.

Although, the behavior of $I_i(\underline{f})$ functions is not analytically studied in this research, experimental results, presented in Fig. 1, show that $I_i(\underline{f})$ is a well-behaved function with respect to f_{i+1} value, when f_i and r_i values are known. It is worthy to note that the values of f_i and r_i are previously known at each step, and we aim to find the value of f_{i+1} which makes $I_i(.)$ zero. Therefore, finding the root of $I_i(.)$ corresponds to find the f_{i+1} value given known values of f_i and r_i. In Fig. 1, showing $I_i(\underline{f})$ for some different values of i, the horizontal axis is the difference between f_{i+1} and f_i and the vertical axis is the value of I_i. It is seen that $I_i(\underline{f})$ starts from a positive value and goes to negative values after crossing $I_i = 0$. We have observed, zero crossing of $I_i(\underline{f})$ functions usually occur for a little difference between f_{i+1} and f_i, i.e., starting from $f_i = f_{i+1}$ and going on, we will soon reach the value of f_{i+1} which is the root of $I_i(\underline{f})$ and is located at a little distance from the starting point (i.e. f_i). Hence, to find the root of each $I_i(\underline{f})$ in (4), we use a simple search method in which f_{i+1} starts from f_i and after a small number of increments, the root will be met. Another property of $I_i(\underline{f})$ functions is that the difference between f_i and f_{i+1} decreases with i. This can help to limit the cost of root finding.

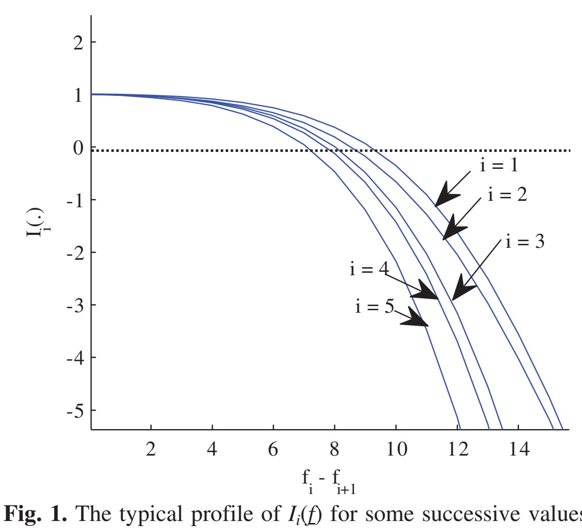


Fig. 1. The typical profile of $I_i(\underline{f})$ for some successive values of i, the horizontal axis is the difference between f_{i+1} and f_i. In this figure, $N = 200$, $L = 47$, and $p = 0.2$. The similar profiles will be obtained for other values of these parameters.

The value of f_1 can not be found using (4). So, in general case, for different values of f_1, the corresponding $\underline{f}$ should be found by the step-by-step progressive process, and D_{avg} should be calculated using (1). Then, the f_1 and the corresponding $\underline{f}$ suggesting the minimum D_{avg} are selected as the best solution. However, an approximation method is used to reduce the number of iterations for finding the optimal value of f_1, which will be discussed in the next subsection. The proposed optimization process can be summarized as in Fig. 2.

```
Compute f*1 using (7)
r0 = 0;
for f1 = f*1 - h/2 to f*1 + h/2{
  r1 = N - f1;
  for i = 1 to L - 1{
    fi+1 = fi;
    ri+1 = ri + N - fi+1;
    Compute Ii using (4)
    while (Ii ≠ 0 and fi+1 >= 0){
      fi+1 = fi+1 - 1;
      ri+1 = ri+1 + 1;
      Compute Ii using (4)
    }
  }
Compute Davg using (1)
Store Davg along with f = (f1, .., fL)
}
Select f with the least Davg
```

Fig. 2. The pseudo code of the proposed method to find the optimum $\underline{f}$. The parameter h represents the length of search interval of f_1 around the f^*_1, usually $h = 10$ is appropriate.

3.1. Approximation of Optimal f_1 value

So far, we have derived a progressive relation between the sizes of any two consecutive segments. It means, if we have f_1, then we can obtain f_2, then f_3 and etc. In this way, each possible value of f_1 (from 0 to N-1) can initiate a progressive procedure which results in a candidate solution of $\underline{f}$. Among all possible resultant solutions, we should select the one that minimizes the expected distortion in (1). To keep the complexity of the proposed progressive method low, it is necessary to strictly confine the search interval of f_1 as short as possible. In this subsection, we find a short search interval around the optimal value of f_1.

Making zero the derivation of the expected distortion in (1) with respect to f_1, we obtain

$$c'(f_1)[d(N-f_1)-d(0)] = \sum_{j=1}^{L} P_j(\underline{f})d'(r_j) \tag{5}$$

Using (5), we can find the optimal value for f_1 provided that all other f_j values, j = 2, …, L, are known. To obtain a good approximation of the optimal f_1 value from (5), we set $f_2 = f_3 = \ldots = f_L = f_r$, where f_r is the rate-optimal solution [1], which is the solution that maximizes the expected number of received source bits and is simply expressed as [1]

$$f_r = \arg\max_{i=0,\ldots,N-1} (N-i)\sum_{n=0}^{i} p_n \tag{6}$$

In this way, $P_2 = P_3 = \ldots = P_{L-1} = 0$. Usually, we can consider $d'(r_L) \approx 0$, because r_L is sufficiently large. Therefore, (5) is reduced to

$$c'(f_1)[d(N-f_1)-d(0)] \approx (c(f_1)-c(f_r))d'(N-f_1) \tag{7}$$

Let the solution of (7) for f_1 is f^*_1 which can be found with a simple search. Afterwards, we can say the optimal f_1 value is located in a short interval around f^*_1. To validate this claim, several simulations have been carried out for various selections of parameters set, including N, L, $c(.)$, and $d(.)$. Simulation results show that $[f^*_1 - h/2, f^*_1 + h/2]$ is an appropriate interval for nearly all natural cases, in which h is at the order of 10 (albeit, h is dependent to N and L values to some extent, but not severely).

If the search interval for f_1 consists of h points and root finding search of each $I_i(\underline{f})$ runs for an average of b times (successive increments of starting f_i) until the solution of (4) is found, then we can roughly say that the complexity of the progressive method is $O(hbL)$. This means that the complexity of the proposed method does not essentially depend on N and this is a great advantage, especially for large values of N. As mentioned earlier, the local search algorithm has a complexity of $O(NL)$ [1] which is linearly dependent to N.

4. SIMULATION RESULTS

In this section, we compare the performance of our progressive algorithm with the local search algorithm in [1], which is approved as an efficient and fast algorithm to solve the ULP packetization problem. In [1], some of the other well-known approaches have been compared with the local search algorithm in the sense of simplicity and efficiency and the local search algorithm has been recognized as the best one. Hence, we compare our method only with the local search algorithm. Although the proposed and local search methods are both applicable on image and video sources, we compare them for image coding application. To generate an embedded bit stream from a test image, we have used JPEG2000 scalable coding. Obviously, the comparison can be easily extended to scalable video codecs, e.g. 3D set partitioning in hierarchical tree (3D SPIHT), with a known distortion-rate characteristic.

In our simulation, the packet loss PDF is considered as a binomial distribution with parameters p and N. It means each of N packets may be independently lost with the probability of p. Because ATM packets have a payload length of 48 bytes and one byte is required for sequence number, we have chosen L = 47 bytes to mimic a practical application. JPEG2000 scalable bit streams of 8 bits per pixel (bpp) gray scale 512×512 Lena, Barbara, and Boat test images are generated with the Jasper 1.701.0 transcoder [5]. The fidelity of the reconstructed image is measured in peak signal-to-noise ratio (PSNR). The PSNR-rate curves of JPEG2000 coded Lena and Barbara images are shown in Fig. 3.

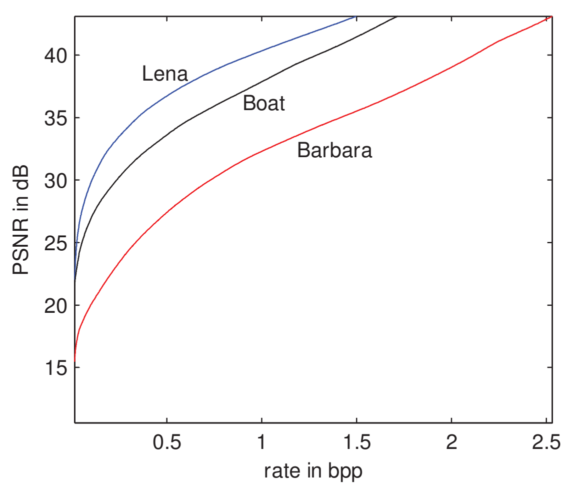
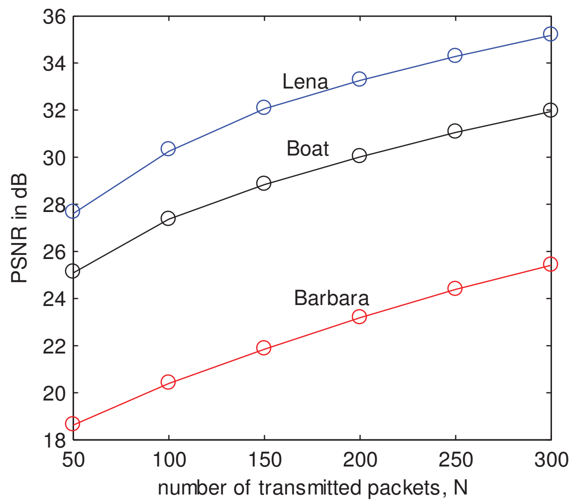
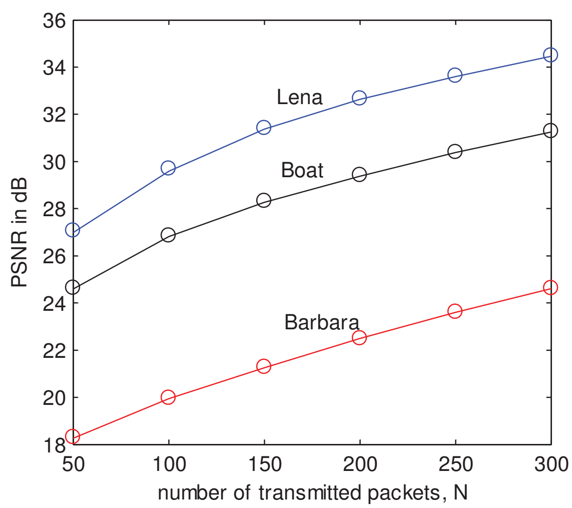


Fig. 3. The PSNR-rate curves of JPEG2000-coded 8-bpp gray scale 512×512 Lena, Boat, and Barbara images.

The rate after compression and MD-FEC is 8*NL* / (512×512) in bpp. Therefore, N value can be used as a rate control parameter. In Fig. 4, PSNR performance of the proposed progressive method is compared with the local search algorithm for some values of N. The packet loss probability, p, is assumed 0.1 and 0.2 for Fig. 4a and Fig. 4b, respectively. It is seen that the both ULP segmentation methods have closely the same performances.



(a)



(b)

Fig. 4. The comparison of the proposed method (solid lines without any marker) with the local search algorithm [1] (circles), (a) PSNR vs. N for $p = 0.1$, (b) PSNR vs. N for $p = 0.2$.

5. CONCLUSIONS

In this paper, we have developed a low complexity progressive method to determine the suboptimal size of each data segment in an embedded code stream for the purpose of unequal loss protected-packetization. The proposed method is based on an analytically-derived relation between the sizes of any two successive segments. To keep the number of required progressive procedures low, a short search interval including the optimal size of the first segment has been derived analytically.

The search space of the segmentation problem for unequal loss protected-packetization is a large and "dark" space and none of the earlier researches have focused on the behavior of the expected distortion in such a dark space. The proposed method is based on a smart analysis of the expected distortion in the search space. The simulation results show that the performance of the proposed method is similar to performance of the local search algorithm. Although the proposed method does not outperform the well-known efficient algorithm of local search developed in [1], but there is a kind of novelty in developing a new concept; "In the optimal point of the expected distortion, there is a unique relation between the sizes of any two consecutive segments." This noticeable concept leads to this fact that if we have the optimal size of the first segment, then we can find the sizes of the rest of the segments by means of a simple progressive procedure. This means it is not needed to find the whole segments sizes jointly, but we can apply a kind of separation between them. This fact is a "brightening torch" to confidently step towards the optimal point of the expected distortion curve in a completely dark and huge search space. Hence, this analysis style may be used as a basis for further developments and improvements in the future.

The complexity of the proposed method is nearly independent of the number of packets and this is a great advantage, especially when a large number of packets should be transmitted. The complexity of the local search algorithm is linearly dependent to the number of packets.

6. REFERENCES

[1] V. Stankovic, R. Hamzaoui, and Z. Xiong, “Packet loss protection of embedded data with fast local search,” Proc. *IEEE ICIP-2002,* Rochester, NY, Sept. 2002.

[2] S. Dumitrescu, X. Wu, and Z. Wang, “Globally optimal uneven error-protected packetization of scalable code streams,” Proc. *DCC'02*, Snowbird, Utah, Apr. 2002.

[3] R. Puri, K. Ramchandran, “Multiple description coding using forward error correction codes,” in *Proc. 33rd Asilomar Conf. on Signals and Systems*, Pacific Grove, CA, Oct. 1999.

[4] A. E. Mohr, E. A. Riskin, and R. E. Ladner, “Unequal loss protection: graceful degradation of image quality over packet erasure channels through forward error correction,” in *IEEE Journal on Selected Areas in Comm*, vol. 18, no. 1, pp. 819-828, 2000.

[5] Available at http://www.ece.uvic.ca/~mdadams/jasper

[6] J. Peng, C.-S. Kim, and C.-C. J. Kuo, “Technologies for 3D mesh compression: A survey,” *Journal of Visual Communication and Image Representation*, vol. 16, pp. 688–733, Dec. 2005.

[7] A. Norkin, M. O. Bici, G. Bozdagi Akar, A. Gotchev, and J. Astola, “Wavelet-based multiple description coding of 3-d geometry,” in *VCIP 2007*, San-Jose, US, Jan. 2007.

[8] G. AlRegib, Y. Altunbasak, and J. Rossignac, “An unequal error protection method for progressively transmitted 3-D models,” *IEEE Trans. Multimedia*, vol. 7, pp. 766–776, Aug. 2005.

[9] S. Ahmad and R. Hamzaoui, “Optimal error protection of progressively compressed 3d meshes,” in *Proceedings IEEE InternationalConference on Multimedia and Expo (ICME’06)*, Jul. 2006.