EURASIP Journal on
Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

Optimization for image transmission over varying channel with MCMC

Xiaobin Wu, Lei Cao* and Paul Goggans

Abstract

Existing work in media transmission generally assumes that the channel condition is stationary. However, communication channels are often varying with time in practice. Adaptive design needs frequent feedback for channel updates, which is often impractical due to the complexity and delay. In this article, we design the unequal error protection for image transmission over noisy varying channels based on their distribution functions. Since the channel effect must be marginalized in order to find the appropriate rate allocation, the optimization problem is very complex. We propose to solve this problem using the Markov Chain Monte Carlo (MCMC) method. The cost function is first mapped into a multi-variable probability distribution. Then, with the "detailed balance", MCMC is designed to generate samples from the mapped stationary probability distribution so that the optimal solution is the one that gives the lowest data distortion. We also show that the rate allocation design considering the channel probability function works better than the design considering the mean value of the channel.

Keywords: Progressive image transmission, Optimal rate-allocation, Markov chain monte carlo

Introduction

Progressive image compression, such as SPIHT [1], is an approach that exploits the inherent similarities across the sub-bands in a wavelet decomposition of an image, and the algorithm codes the most important wavelet coefficients first, and transmits the bits so that an increasingly refined copy of the original image can be obtained progressively. The progressive compression is widely used in many applications, because the media can be restored with the best quality by receiving a sequence of continuous error-free data. However in the coded data stream, any error bit due to channel noise would cause the loss of synchronization between the sender and receiver, which means that all the data after that bit error has to be completely discarded. Therefore, an important issue in image transmission is to design a protection strategy for the source data, i.e., allocating channel code rates to different data packets, based on the channel condition and the rate-distortion feature of the source, in order to optimize the overall recovery quality of image in the noise channel.

In [2], the cyclic redundancy check codes and rate compatible punctured codes (CRC/RCPC) were employed to protect the SPIHT coded data and obtained performance better than previous results in binary symmetric channels (BSCs). This work used equal error protection and was then extended to the product code protection [3] when the Gilbert–Elliot channel (GEC) model was considered. Since then, many error-control solutions for progressive image transmission [3-11] have been proposed. In these methods, different codes are used, including CRC/RCPC [5,7,8], CRC/RCPT codes [6,9-11], Reed-Solomon (RS) codes [4,11] and their product codes. Different channel conditions are also considered, including BSC, GEC, and packet loss channels.

All these methods consider only the fixed channel condition. In practice, however, the channel condition always changes with the time, mainly due to the mobility and multi-path factors in the communication. This feature of channels has also been considered for a while in video transmission, such as [12] where the scalability of the precoded JPEG 2,000 video is considered and a fast source rate allocation through the steepest descent algorithm is designed. In this type of work, the varying feature of channel is often represented as different transmission bandwidth. The physical additive white Gaussian noise

*Correspondence: lcao@olemiss.edu
Department of Electrical Engineering, University of Mississipp, Oxford, MS 38655, USA

(AWGN) and channel coding are generally not considered. In this article, we consider the unequal error protection (UEP) design through channel codes for varying channels. Intuitively, one designed UEP system has to be updated with frequent feedback of the channel condition. However, this is impractical as the multi-dimensional optimization process for the rate allocation design is often very complicated and the channel variation can be fast. Therefore, a more realistic solution is to design the protection method considering all the channel conditions. The channel variation, which is captured by the probability density distribution (PDF) of signal to noise ratio (SNR), can often be estimated from a long period of operation of the communication network. Considering the channel PDF, the optimization problem is even more complicated than previous designs because channel effects must also be marginalized.

In this article, we apply the Markov Chain Monte Carlo (MCMC) technique to the optimal UEP system design where varying channels are considered. MCMC is a method of sampling from probability density function based on constructing a Markov chain. With the "detailed balance" mechanism, the equilibrium states of the Markov chain describe the targeted probability distribution [13]. Since the purpose is to find out the optimal channel code rate allocation so that the transmitted image has the lowest distortion in the varying channel, we first use the exponential function of the simulated annealing method to map the cost function into a probability distribution. Then we use the slice sampling of MCMC to generate samples. Since these samples drawn from MCMC approach the mapped stationary probability distribution, the MCMC method not only suggests the optimal design but also provides more probabilistic information than other heuristic optimization methods. We show that the MCMC method enables a low complexity design with a solution approaching the optimal one. Finally, we show that the system design based on the channel distribution outperforms a design that is based on a specific channel statistic such as the mean value.

The rest of this article is organized as follows. In Section "Problem description", The image transmission problem is described and formulated. In Section "MCMC method with slice sampling", the MCMC method and how it is applied to the image transmission problem is discussed. Simulation results are presented in Section "Simulation and results" and the conclusion is drawn in Section "Conclusion".

Problem description

We consider a joint source-channel coding system with N coded packets with fixed packet length as shown in Figure 1. This data structure was considered in [6] and is justified in many applications. Rate compatible punctured turbo (RCPT) codes are used to provide UEP. Assume that the space of different channel code rates is $\mathcal{C} = \{c_1, c_2, c_3 \ldots, c_M\}$ with $c_1 > c_2 > \cdots > c_M$ and $C_i, i = 1, \ldots, M$ represents the rate of the length of source data and the length of packet. Since each packet has the fixed length L, the corresponding number of source bits for each code rate is $s_i = c_i \cdot L$. Therefore the number of the overall source bits is $\sum_{i=1}^{N} c_i \cdot L$ and the number of overall channel parity bits is $\left(1 - \sum_{i=1}^{N} c_i\right) \cdot L$. After being transmitted over a specific channel with SNR $= x$ dB, the packet error probabilities after decoding are denoted as $\mathcal{E} = \{e_1(x), e_2(x), \ldots, e_M(x)\}$, where $e_1(x) > e_2(x) > \cdots > e_M(x)$ [14].

In this article, the Turbo code $(15, 17)_{\text{oct}}$ with mother code rate of 1/3 is considered. A set of code rates of $\{4/4, 4/5, 4/6, 4/7, 4/8, 4/9, 4/10, 4/11, 4/12\}$ are obtained through puncturing [15]. Coded packet size is $L = 4096$. Figure 2 shows the residual packet error rate (PER) in different AWGN channel conditions. It needs to be noted that the PER also depends on the packet size and can be different if a different packet length is used. A larger data length increases the coding gain in turbo decoding in general.

Suppose that a message contains N coded packets, each being protected with channel code rate of $r_i \in \mathcal{C}, i = 1, \ldots, N$. N is determined by the total transmission rate and the packet length. For channel x, the corresponding PER is $p_i(x) \in \mathcal{E}$. Denote $P_i(x)$ as the probability that the

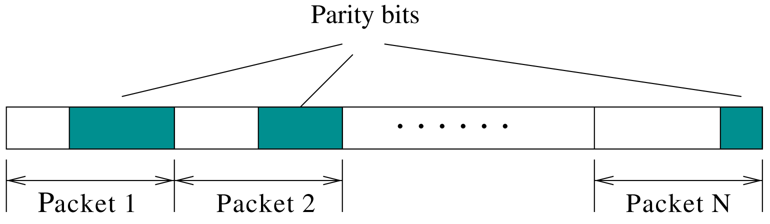

Figure 1 Data organization. A message via channel contains *N* packets. Each packet could contain different length of parity part and data part, but each packet has the same length of bits.

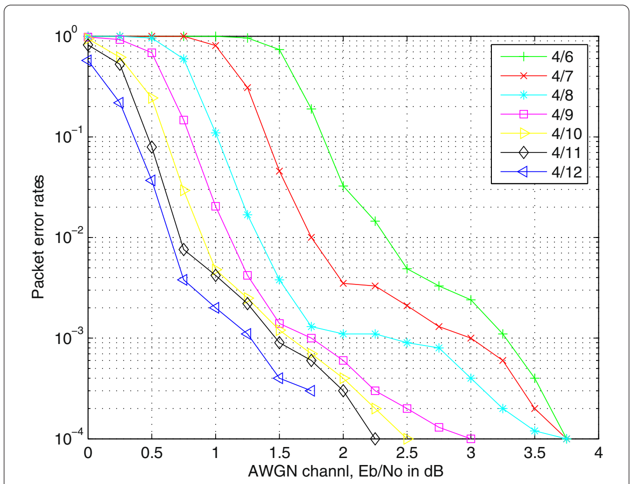

Figure 2 Turbo code performance for fixed coded packet length. PER in AWGN channels with different SNR values [16].

first i packets are decoded without errors but the $i + 1$th packet is not correctly decoded. Then

$$P_i(x) = \begin{cases} p_1(x) & \text{if } i = 0 \\ \Pi_{j=1}^{i}(1 - p_j(x))p_{i+1}(x) & \text{if } i = 1, \ldots, N-1 \\ \Pi_{j=1}^{N}(1 - p_j(x)) & \text{if } i = N \end{cases} \tag{1}$$

Distortion between the recovered image and the original image is represented by the mean square error (MSE). Let D_i be the distortion of the image restored by the first i packets that have been correctly decoded. Then, with an allocation $\mathcal{R}_N = \{r_1, r_2, \ldots, r_N\}, r_i \in \mathcal{C}$, of channel code rates for the data packets, the expected distortion is

$$D_N(x, \mathcal{R}_N) = \sum_{i=0}^{N} P_i(x) D_i \tag{2}$$

This distortion in Equation (2) has a recursive representation [5] as:

$$\begin{aligned} D_N(x, \mathcal{R}_N) &= D_0 - E_N(x, \mathcal{R}_N) \\ E_N(x, \mathcal{R}_N) &= (1 - p_1)[\,\Delta D_1 + E_{N-1}(x, \mathcal{R}_{N-1})] \end{aligned} \tag{3}$$

where D_0 is the distortion of the case that no correct packet is received. ΔD_i is the reduced distortion between D_{i-1} and D_i. $\mathcal{R}_{N-1} = \{r_2, \ldots, r_N\}$ is the rate allocation for the last $N - 1$ packets.

When the channel SNR is given, a dynamic programming method was suggested in [5] for this distortion-based optimization problem. This method is a backwards updating process. That is, the reduced distortion of a latter packet should be used first to determine the optimal allocation. However, the reduced distortion in a latter packet actually depends on the channel code rates in former packets, which is evident in Equation (3). Therefore, although the forward-updating process along trellis is more practically used [6,11], the solution becomes sub-optimal as indicated in [6].

For a varying channel, if the channel SNR density function $f_X(x)$ is known, then the joint source-channel coding problem becomes to find a channel code rate set $\mathcal{R}_N$ which minimizes, subject to a given overall transmission rate,

$$\begin{aligned} D_N(\mathcal{R}_N) &= \int_{-\infty}^{\infty} f(x) D_N(x, \mathcal{R}_N) dx \\ &= \int_{-\infty}^{\infty} f(x) \sum_{i=0}^{N} P_i(x) D_i dx \end{aligned} \tag{4}$$

Apparently, this is different from finding the optimal rate-allocation based on the mean channel condition, because the mean SNR is a fixed value and the distortion is

$$D_N(\bar{x}, \mathcal{R}) = \sum_{i=0}^{N} P_i(\bar{x}) D_i \tag{5}$$

where $\bar{x}$ is the mean channel SNR. The varying channel case is more complicated than considering only on the specific channel instance as the channel variable x must be marginalized in the optimization process. In next section, we propose to solve this optimization problem with the MCMC method.

MCMC method with slice sampling

In this section, we address the mapping between the cost function and the stationary probability used in MCMC, the technique of slice sampling method and their application to the image transmission problem.

Mapping the cost function to probability

The cost function has the channel code rate of each packet as input, and the distortion value as output. When the image message is transported by N packets, there are N inputs, i.e., $\mathcal{R}_N = \{r_1, \ldots, r_N\}$, need to be considered to generate the output. The optimization task is to find the $\mathcal{R}_N$ that gives the minimum distortion value. The first issue in using MCMC is the need to map the cost function $J(\cdot)$ into a likelihood function that could be used as the stationary probability distribution in the Markov Chain. We use the function in the simulated annealing for this purpose, i.e.,

$$\frac{1}{Z_\tau} \exp\left\{-\frac{J(\cdot)}{\tau(t)}\right\} \tag{6}$$

where Z_τ is a normalization constant so that any possible value is always between 0 and 1 and the sum of all probability is 1.

If the current state of the Markov chain $x(t)$ is equal to i, suppose a neighbor j of i is visited with a probability q_{ij} where $q_{ij} = q_{ji}$ is required. Then whether j will be selected as a new sample is given by the probability of

$$\min\left\{1, \exp\left[-\frac{J(j) - J(i)}{\tau(t)}\right]\right\}. \tag{7}$$

This is termed as the Metropolis process and it can be proved that with this process, the stationary probability of the Markov Chain is (6). It can be noted that

(1) If $J(j) \to \infty$, then $\exp\{-[J(j) - J(i)]/\tau(t)\} \to 0$. In this case the accepting probability for j is tending to 0. That is, a state j associated with a high cost has a low chance to be accepted.
(2) On the other hand, that $J(j) \to 0$ means the new cost is negligible. Then $-[J(j) - J(i)]/\tau(t)$ could be positive, which implies the new state j is always selected due to its lowest cost value.

The parameter $\tau(t)$, termed as the temperature in the simulated annealing, also needs to be considered carefully. To make the algorithm approach the optimal solution, the temperature should change with the time. The temperature function is a non-increasing function, which represents the cooling down process. $\tau(0)$ is usually set as a high value in the beginning of simulation. This makes sure that with a high probability a new state will be accepted. In this stage, system may reach more diverse and different states based on their probability contribution. When $\tau(t)$ approaches 0, the probability becomes very small. The system is reluctant to accept a new state j, which is often described as a "Frozen" stage. The process of decreasing of temperature is usually slow to give the system enough time to be "trapped" in the low cost area. In our simulation, the temperature value is reduced after a new sample is generated.

Slice sampling

Simulated annealing uses the normal Metropolis Hastings method [17] for sampling from the probability distribution. The method is based on the observation that to sample a random variable one can sample uniformly from the region under the graph of its density function. Compared with the slice sampling [13], a normal Metropolis method is inefficient for two reason. First, with a fixed step size, the simple Metropolis method may not characterize well the local property of the probability function. Second, the simple Metropolis method takes a random walk to find next step, and therefore it could take a large number of steps to reach a state which is only a few steps away. The slice sampling can be applied to wherever the Metropolis method can be applied to. It has the advantage over the simple Metropolis method in that it is more flexible to the update of parameters like the step size, and the only requirement is that the target density function $P(x)$ can be evaluated at any state $x(t)$[18]. A brief description of the slice sampling in one dimension is as follows.

```
(1) evaluate $P[x(t) = i]$
(2) draw a vertical coordinate,
        $u \sim \text{Uniform}(0, P[x(t) = i])$
(3) create a neighbor space enclosing current state $i$,
        $i \in (I_{\text{left}}, I_{\text{right}})$
(4) start loop {
(5)     draw $j$ from neighbor space, $j \in (I_{\text{left}}, I_{\text{right}})$
(6)     evaluate $P[x(t+1) = j]$
(7)     if $P[x(t+1) = j] \geq u$, break out loop and
            the next state $x(t+1) = j$ is accepted
(8)     else modify the neighbor space and repeat loop
(9) }
```

$P[x(t) = i]$ is the probability function. $x(t) = i$ represents that at time t, the state of system is i. The time is indexed as $t = \{1, 2, 3, \dots\}$.

An important improvement over the simple Metropolis method here is the "create and modify the neighbor space" in steps 3 and 8. Based on these two steps, slice sampling could change region of neighbor space to locate the next possible state in order to accelerate the process of finding a new sample. Description of steps 3 and 8 in the one dimension case is as follows,

Step 3: creating the neighbor space of one dimension problem

```
(1) set a value of $w$ as the "width"
(2) generate rand $\sim$ Uniform$(0, 1)$
(3) $I_{\text{left}} = i - \text{rand} * w$
(4) $I_{\text{right}} = i + \text{rand} * w$
(5) while $P[I_{\text{right}}] > u$, then $I_{\text{right}} = I_{\text{right}} + w$
(6) while $P[I_{\text{left}}] > u$, then $I_{\text{left}} = I_{\text{left}} - w$
```

where w is a constant set in the beginning by user, which is used to represent the region that the state could go each time. The $(I_{\text{left}}, I_{\text{right}})$ is the neighbor space of state i. In the "creating" process, the slice sampling uses two loops to find the adaptive region of neighbor space. As we can see from step 3, it is actually a process to extend the region of possible neighbor of current state.

Step 3 can also be understood by Figure 3 and its explanation. Too large a region in Figure 3 would cause high computational complexity. Therefore, it also needs a strategy to shrink the region again, which is implemented in step 8. Whenever a new state $x(t+1) = j$ is rejected, the method tries to "modify" the neighbor space.

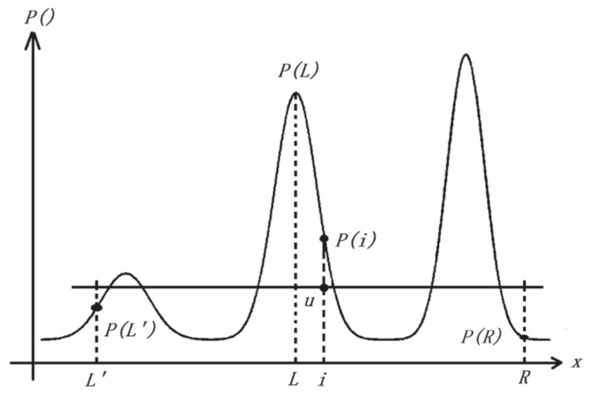

Figure 3 Description of slice sampling. $[L, R]$ is the neighbor space enclosing current state i. Left end of the interval $P(L)$ is evaluated and is known to be greater than u, so a step to the left of size w is made, which make left end to L'. Check $P(L')$ which is smaller than u, then stop and step out. Also at the right end of the interval, $P(R)$ is evaluated and is smaller than u. So no stepping out is needed. When the above process it done, values of P at both ends should be smaller than u. Since the neighbor space is decided as $[L', R]$, the next state candidate will be picked randomly in the interval $[L', R]$.

Step 8: modifying the neighbor space of one dimension problem:

(1) if $j > i$, then $I_{\text{right}} = j$
(2) else $I_{\text{left}} = j$

Note that comparing $j > i$ is to find out the relative position of the states and then shrink the neighbor space. In one dimension, i, j correspond to the values of x-axis. In multi-dimension, values along difference axises need to be compared separately. In summary, by the strategy of "creating and modifying", slice sampling could adjust the neighbor space. In addition, by defining value of w, we may also control the speed of stepping out and shrinking of the space.

Since the problem considered is a multiple-dimensional problem, the previous one-dimension slice sampling should be adjusted to handle the multi-dimensional case. Consider the input at time t with a set of parameters:

$$\text{Input}(t) = \{x_1(t), x_2(t), x_3(t), \ldots, x_N(t)\}.$$

One way is that only one variable is changed at each time in a given order. Let $x_1(t)$ be selected at time t. The method will find out the next state based on $x_1(t)$ with all other variables unchanged. Then, we fix $x_1(t)$ and find the next state based on the variation of $x_2(t)$. This process continues when enough samples are drawn. Alternatively, we may also change all variables simultaneously in any specific time instant. An interesting view of this method is considering each variable of input as a direction that the system could head to. By combining all directions, we know which state the system should move to. In this article, the second method with parallel processing is used, which compared with the first method, has better time convergence performance when generating the same number of samples.

MCMC for UEP in image transmission

The distortion function is the cost function. The goal is to find the the appropriate channel code rates for different packets so that the lowest distortion can be obtained for image transmission. We assume that the channel SNR is in the range of $[0, 4]$ dB as shown in Figure 2. The turbo code rate is one of the 9 different numbers in the set of

$$\mathcal{C} = \{4/4, 4/5, 4/6, 4/7, 4/8, 4/9, 4/10, 4/11, 4/12\}$$

$\mathcal{R}_N$ is the input of the cost function. It is a vector with N elements, each representing the code rate for each packet, i.e., $\mathcal{R}_N = \{r_1, r_2, \ldots, r_N\}$, and $r_i \in \mathcal{C}$. Apparently, our simulation Markov Chain has the probability distribution like

$$P[x(t) = \mathcal{R}_N] = \frac{1}{Z_\tau} \exp\left[-\frac{D_N(\mathcal{R}_N)}{\tau(t)}\right]$$

where Z_τ is a normalizing constant. Because the probability values are only compared in the slice sampling, we can simply set $Z_t = 1$. The method used is actually a combination of simulated annealing and slice sampling. We use the probability representation of simulated annealing and reduce the temperature with time, while using the slice sampling to access new states and generate new samples.

The MCMC process is the same as steps described in the previous section. Since there are multiple input corresponding to different frames, some modifications need to be considered in the following.

Step 1: The state $x(t) = i$ becomes $x(t) = \mathcal{R}_N$ in the MCMC process. $P[x(t) = \mathcal{R}_N]$ is a function for the input $\mathcal{R}_N$.

Table 1 Rates corresponding to numbers

Rate	Represent
4/4	1
4/5	2
4/6	3
4/7	4
4/8	5
4/9	6
4/10	7
4/11	8
4/12	9

Table 2 Distortion and frequency from the brute force search: uniform/ten packets

Distortion	Frequency
220.7	1
220.8	2
220.9	5
221.0	9
221.1	13
221.2	27
221.3	34
221.4	40
221.5	68
221.6	103

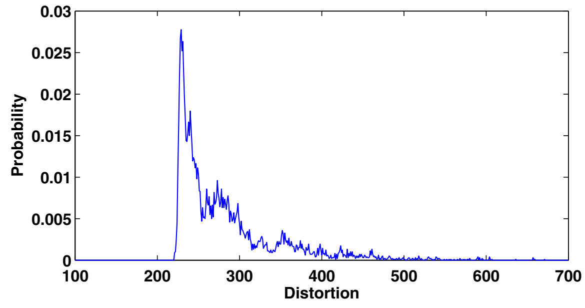

Figure 5 Simulation result with reducing temperature, uniform distribution, ten packets. The probability distribution is the histogram of different distortion value divided by the total number of samples.

Step 3: for $\mathcal{R}_N = \{r_1, r_2, \ldots, r_N\}, r_i \in \mathcal{C}$, constructing an interval space for each variable r_i. That makes the neighbor space look like:

$$\text{space} = \{(r_1^{\text{left}}, r_1^{\text{right}}), (r_2^{\text{left}}, r_2^{\text{right}}), \ldots, (r_N^{\text{left}}, r_N^{\text{right}})\}$$

Our simulation uses parallel process to find the neighbor space. That is, the neighbors of all r_i,$i = 1, \ldots, N$, are updated before the next state is drawn and checked.

```
(1)  set a value for $w_i$
(2)  loop $(r_1 \rightarrow r_N)\{$
(3)      generate rand $\sim$ Uniform$(0, 1)$
(4)      $r_i^{\text{left}} = r_i - \text{rand} * w_i$
(5)      $r_i^{\text{right}} = r_i + \text{rand} * w_i$
(6)      check $r_i^{\text{left}}, r_i^{\text{right}} \in \mathcal{C}$
(7)      while $P[\{r_1, \ldots, r_i^{\text{left}}, \ldots, r_N\}] > u$, then we have
       $r_i^{\text{left}} = r_i^{\text{left}} - w_i$
(8)      while $P[\{r_1, \ldots, r_i^{\text{right}}, \ldots, r_N\}] > u$, then we have
       $r_{\text{right}} = r_{\text{right}} + w_i$
(9)      check $r_i^{\text{left}}, r_i^{\text{right}} \in \mathcal{C}$
(10) $\}$
```

The process is same as the one-dimensional case, except that a *loop* is needed here because the neighbor space needs to be checked for each $r_i \in \mathcal{R}_N$. In addition, different w_i may be used if necessary. For example, in some cases the input may have different range. A common practice is to set w_i as a certain percentage of the entire range of the parameter space.

Step 8: Let $\mathcal{R}'_N = \{r'_1, \ldots, r'_N\}$ be the new state, then the neighbor space is modified as

```
(1)  loop $(r_1 \rightarrow r_N)\{$
(2)      if $r'_i > r_i$, then $r_i^{\text{right}} = r'_i$
(3)      else $r_i^{\text{left}} = r'_i$
(4)  $\}$
```

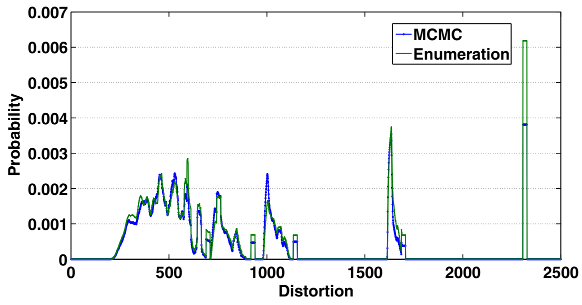

Figure 4 MCMC versus enumeration: uniform distribution, ten packets. The histograms of all possible distortion based on both MCMC and enumeration are collected. The probability distributions are obtained by dividing each histogram by the total number of samples generated by MCMC and enumeration, respectively. Comparing both curves shows that the MCMC method indeed draws samples from the mapped distribution.

Table 3 MCMC result: uniform distribution, 32 packets

Input rate	Distortion
999999988888877765555444444332211	49.71207
9999999888888777666655554444332211	49.43509
9988888888877777666655555544333321	48.06369
999999988888887777666555554443331	46.64615
9999888888888777777666665554444431	46.55235
999888777777776666666555444444333	46.49173
999999988888887777666665555544443	45.31980
999999888777666666655555555554443	45.28976
999999988888877776666655555554444	45.19252
999999988888877776665555555554443	45.19155

Table 4 Performance based on different bpp in the range of SNR =[0, 4] dB ∼ $\mathcal{N}$(2, 0.5)

Packets number	Optimal solution	Overall bpp	Source code bpp	Protect code bpp
10	9876555544	0.1563	0.0732	0.0831
16	9998877655 555444	0.25	0.1123	0.1377
32	9(6)8(6)7(4)6(3)5(9)4(3)3(1)	0.50	0.2187	0.2813
48	9(10)8(7)7(9)6(5)5(10)4(6)3(1)	0.75	0.3249	0.4251
64	9(21)8(10)7(5)6(11)5(10)4(6)3(1)	1.00	0.4159	0.5841

Since the code rate space is $\{4/4, 4/5, 4/6, \ldots, 4/12\}$, which contains nine possible values, instead of multiplying w, a random integer, rand $\sim$ Rand$(0, w)$, could be applied as $r_i^{\text{right}} = r_i + \text{rand}$ in simulation.

Another issue is the set of the temperature function $\tau(t)$. In any MCMC simulation, an adaptive $\tau(t)$ is an important factor for the performance. While with big $\tau(t)$, system would take very long time to reach convergence, with very small $\tau(t)$, the system may be trapped in a local minimum. A reasonable $\tau(t)$ is necessary. In [19] Hajek gave a popular temperature function:

$$\tau(t) = \frac{d}{\log(t)}$$

where d here is a constant number which is considered as the measurement of how difficult for the system $x(t)$ to jump out from the current local minimum and travel to the optimal solution set. In other words, MCMC needs a large enough d to start off but reduces this value with the running of the chain.

Simulation and results

In this section, the MCMC method has been applied to design the UEP system for progressive image transmission. The Lena image with size of 512×512 and 8 bits per-pixel (bpp) was passed through the SPIHT algorithm to obtain the source data. RCPT codes are then used for channel protection, where a set of numbers are used to represent the set of channel coding rates, as shown in Table 1. The distortion value is the MSE between the reconstructed image and the original image.

We first consider a case with a small number of packets, with the purpose to check whether the MCMC method can perform as expected. Then we consider cases with more number of packets and different channel probability distributions to study the performance based on the MCMC method. We also compare the design results based on the channel distribution and the mean value of the channel.

MCMC samples in the case of ten packets

The channel distribution is uniformly distributed, i.e., $f_X(x) \sim \text{Uniform}[0, 4]\, dB$. MCMC is used to generate the samples of $\mathcal{R}_N$ based on the probability mapped from the cost function. Different trials are conducted, i.e., the MCMC process is repeated to find independent UEP designs. It was found that the MCMC almost always gives the optimal $\mathcal{R}_N$. It closely approaches the optimal design even if the optimal result was not reached occasionally.

Table 5 Performance based on channel of SNR =[0, 4] dB ∼ $\mathcal{N}$(2, 0.5) and channel with SNR = 2 dB

Packets number	MSE [0,4] dB	PSNR [0,4] dB	MSE [2] dB	PSNR [2] dB
10	104.0709	27.9575	160.6278	26.0726
16	73.98707	29.4392	88.4346	28.6646
32	45.19155	31.5802	48.3620	31.2858
48	35.13393	32.6735	37.6239	32.3762
64	29.95420	33.3662	30.9719	33.2211

Table 6 Performance based on channel of SNR =[0, 4] dB ∼ $\mathcal{N}$(2, 1) and channel with SNR = 2 dB

Packets number	MSE [0,4] dB	PSNR [0,4] dB	MSE [2] dB	PSNR [2] dB
10	135.9264	26.7978	234.1765	24.4354
16	105.3338	27.9051	136.1534	26.7905
32	76.27627	29.3069	82.2316	28.9804
48	66.13733	29.9263	71.5604	29.5841
64	61.02982	30.2754	62.7087	30.1575

Table 7 Performance based on channel of SNR $=[0,4]$ dB $\sim$ Uniform and channel with SNR $=2$ dB

Packets number	MSE [0,4] dB	PSNR [0,4] dB	MSE [2] dB	PSNR [2] dB
10	220.7620	24.6916	515.7425	21.0065
16	191.3246	25.3131	255.8785	24.0505
32	163.8800	25.9855	176.9067	25.6534
48	154.3131	26.2468	164.3646	25.9727
64	149.5569	26.3827	151.9254	26.3145

Table 8 Cost based on different bpp with $\mathcal{N}(2,1)$ and SNR $=[0,4]$ dB

Overall bpp	Total inputs	Cost ratio
0.1563	9^{10}	1.01e-005
0.25	9^{16}	8.07e-12
0.50	9^{32}	5.68e-27
0.75	9^{48}	3.24e-42
1.00	9^{64}	2.34e-57

To validate the results of MCMC, we exhaustively enumerate all possible $\mathcal{R}_N$ in this case. Brute force method needs to compute all the 9^{10} different rate allocations and takes significant amount of time with a PC. A few least distortion values and the number of corresponding allocations $\mathcal{R}_N$ to achieve each of these values are listed in Table 2. For example, MSE distortion 221.5 has a frequency of 68. In other words, there are 68 different inputs, $\mathcal{R}_N$, that result in the distortion of 221.5. Clearly, the global minimum MSE is 220.7.

To show that the MCMC method really draws samples from the probability function that reflects the distortion, we sample 10,000 samples and plot the histogram together with the results obtained from brute force search. The results are shown in Figure 4. In this simulation, the temperature $\tau(t)$ was set as a large constant value (i.e., 1,000 in simulation) and was not decreased after each sample is generated, which means that the simulation becomes a normal slice sampling method. To make the curves clear in a large x-axis scale, an average function was also used. That is, every x and y-axis value on this figure are the average of the a few neighbors and itself, i.e., $x_i^{\text{new}} = (x_{i-4} + x_{i-3} + \cdots + x_{i+4})/9$. From Figure 4, the results of MCMC and brute force method have very similar shape, which verifies that MCMC indeed generates samples according to the mapped distribution. This property illustrates that the MCMC had visited all regions of the parameter space based on their contribution to the mapped probability.

Figure 5 is a result by reducing temperature by 0.1 starting from 1000, after each new sample is generated. In the figure, the samples are squeezed on the left side. Since the temperature is decreasing when time passes, the samples with small distortion values are more easily picked up. Hence, the histogram is attracted to the left side of x-axis so as to escape from local optimal set.

When the number of packets is 32, brute force method will no longer work as the computational complexity is too high to handle. A few best samples drawn from MCMC are shown in Table 3. Uniform distribution $f_X(x)$ is also assumed for the channel.

Because an early packet loss will cause the discard of packets starting from where the first error occurs, a packet loss in the beginning is crucial to the final performance. The beginning packets should be considered more important than the following packets. One feature that can be observed after hundreds trials is that the rate of each packet in one allocation $\mathcal{R}_N$ is nondecreasing. This result has been theoretically proved in [14,16] for a given channel condition but has not yet been proved for the varying channel case.

Comparison with the case of using channel mean value

Unequal error protection designs based on channel distributions over SNR $=[0,4]$ dB and based on the mean

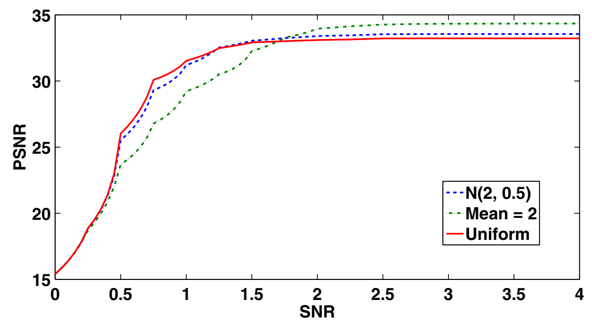

Figure 6 PSNR of optimal $\mathcal{R}_N$, 32 packets, SNR $=[0,4]$ dB. Optimization based on varying channel has better total performance than mean SNR channel.

channel SNR are also compared. Results are showed in the Tables 4, 5, 6 and 7. Three different probability density functions are assumed, which are truncated Gaussian distributions $\mathcal{N}(2, 0.5)$, $\mathcal{N}(2, 1)$ and the uniform distribution used previously. For all these conditions, the mean value of the channel is 2 dB. Using the mean SNR value is a popular choice in existing methods where the design is often based on a specific channel condition.

Table 4 shows the different resource allocation between the source code and the channel code in terms of bpp under different overall bpp cases. The channel distribution is truncated Gaussian $\mathcal{N}(2, 0.5)$. The table also contains the optimal result generated by MCMC where the notation of $a(b)$ denotes b consecutive packets that are coded with code rate a.

Tables 5, 6 and 7 show the distortion comparison for the UEP designs. In each table, the 2nd and 3rd columns are the performance in MSE and peak signal to noise ration (PSNR) of the UEP designs based on channel distribution and the last two columns are the results of using mean SNR for the UEP design. Clearly, these results show that considering all channel distributions does produce better performance than considering a single statistic value of the channel.

In Table 8 we also compare the complexity of MCMC with the case of exhaustive search. Complexity of each case is mainly determined by how many points (i.e., how many different $\mathcal{R}_N$) in the rate allocation space have been tested. The last row shows the ratio of the complexity between MCMC and the exhaustive search. Apparently for $\mathcal{R}_N = \{r_1, r_2, \ldots, r_N\}$, there are 9^N different $\mathcal{R}_N$, since each r_i has nine possible value. The table is generated by counting the time of calculation of MCMC and compared with the number of possible inputs for each bpp. Compared with the exponentially increasing number of possible allocations when more packets are considered, the cost of running MCMC increases linearly and thus results more complexity reduction.

Figure 6 shows the performance of optimal $\mathcal{R}_\mathcal{N}$ generated by the MCMC method for different channel distributions as well as the one generated based on the mean channel value. In the figure, the one based on the mean SNR has lower PSNR range below 1.75 dB approximately, but has better PSNR beyond 1.75 dB. However, the optimal solution $\mathcal{R}_N$ has the overall better average PSNR.

Conclusion

In this article, image transmission over time varying channels is considered. The varying channel, characterized by its probability distribution, makes the optimization problem of finding the best rate allocation very complicated. A MCMC method is proposed to solve this problem. It has been shown that the method can generate near-optimal solutions with low complexity. It also provides an overall picture of the distribution of distortion versus the rate allocation. In addition, it has been shown that the design considering the channel distribution performs better than the design considering the mean value of the channel.

Competing interests
The authors declare that they have no competing interests.

Received: 23 April 2012 Accepted: 30 July 2012
Published: 30 August 2012

References

1. A Said, WA Pearlman, A new, fast, and efficient image codec based on set partitioning in hierarchical trees. IEEE Trans. Circ. Syst. Video Technol. **6**, 243–250 (1996)
2. PG Sherwood, K Zeger, Progressive image coding for noisy channels. IEEE Signal Process. Lett. **4**, 189–191 (1997)
3. PG Sherwood, K Zeger, Error protection for progressive image transmission over memoryless and fading channels. IEEE Trans. Commun. **46**, 1555–1559 (1998)
4. AE Mohr, EA Riskin, RE Ladner, Unequal loss protection: graceful degradation of image quality over packet erasure channels through forward error correction. IEEE J. Sel. Areas Commun. **18**, 819–828 (2000)
5. V Chande, N Farvardin, Progressive transmission of image over memoryless noisy channels. IEEE J. Sel. Areas Commun. **18**, 850–860 (2000)
6. B BA, Robust image transmission using JPEG2000 and turbo-codes. IEEE Signal Process. Lett. **9**(6), 117–119 (2002)
7. V Stankovic, R Hamzaoui, D Saupe, Fast algorithm for rate-based optimal error protection of embedded codes. IEEE Trans. Commun. **51**, 1788–1795 (2003)
8. A Nosratinia, J Liu, B Aazhang, Source-channel rate allocation for progressive transmission of images. IEEE Trans. Commun. **51**, 186–196 (2003)
9. V Stankovic, R Hamzaoui, Z Xiong, Efficient channel code rates selection algorithms for forward error correction of packetized multimedia bitstreams in varying channel. IEEE Trans. Multimed. **6**, 240–248 (2004)
10. V Stankovic, R Hamzaoui, Z Xiong, Fast algorithm for distortion-based error protection of embedded image codes. IEEE Trans. Image Process. **14**, 1417–1421 (2005)
11. N Thomos, NV Boulgouris, MG Strintzis, Wireless image transmission using turbo codes optimal unequal error protections. IEEE Trans. Image Process. **14**, 1890–1901 (2005)
12. L Jimenez-Rodriguez, F Auh-Llinas, MW Marcellin, FAST rate allocation for JPEG2000 video transmission over time-varying channels. IEEE Trans. Multimed. **2012**, 1 (2012)
13. RM Neal, Sampling, Slice. Ann. Stat. **31**(3), 705–767 (2003)
14. L Cao, On the unequal error protection for progressive image transmission. IEEE Trans. Image Process. **16**, 2384–2388 (2007)
15. DN Rowitch, LB Milstein, On the performance of Hybrid FEC/ARQ systems using rate compatible punctured turbo (RCPT) codes. IEEE Trans. Commun. **48**, 948–959 (2000)
16. L Cao, Joint source and channel coding for image transmission over time varying channels. in *IEEE International Conference on Communications, 2007. ICC '07*. (Glasgow, Scotland, 2007)
17. N Metropolis, AW Rosenbluth, Rosenbluth M IN, AH Teller, Teller E, Equation of state calculations by fast computing machines. J. Chem. Phys. **21**(6), 1087–1092 (1953)
18. DJ MacKay, *Information Theory, Inference, and Learning Algorithms*. (Cambridge, Cambridge University Press, 2003)
19. B Hajek, Cooling schedules for optimal annealing. Math. Operat. Res. **1**(2), 311–329 (1988)

doi:10.1186/1687-1499-2012-275
Cite this article as: Wu *et al.*: Optimization for image transmission over varying channel with MCMC. *EURASIP Journal on Wireless Communications and Networking* 2012 **2012**:275.