Determination of Optimal Distortion-Based Protection in Progressive Image Transmission: a Heuristic Approach

Alessio Torquati
University of Genoa-DIST
Department of Communication, Computer and System Sciences
Genoa, Italy
Email: alessio.torquati@dist.unige.it

Maria Fresia
Communications and Remote Sensing Laboratory
Université catholique de Louvain
Louvain-la-Neuve, Belgium
Email: fresia@tele.ucl.ac.be

Fabio Lavagetto
University of Genoa-DIST
Department of Communication, Computer and System Sciences
Genoa, Italy
Email: fabio@dist.unige.it

Abstract—In this paper a method for the fast determination of distortion-based optimal unequal error protection (UEP) of bitstreams generated by embedded image coders is described. The UEP problem is reduced to the more general problem of finding a path in a graph, where each path of the graph represents a possible protection policy and the best path is that one inducing minimal distortion. The problem is combinatorially complex and excludes a brute force approach. The solution is provided by applying heuristic information from the problem domain to reduce search complexity. In particular, we use graph search procedure \textit{A}, well known in the field of artificial intelligence, to avoid exhaustive search. Numerical results show that this technique outperforms the methods presented in \cite{1}, \cite{2}, in terms of Mean Square Error (MSE) distortion and computational complexity.

I. INTRODUCTION

An embedded image coder generates a bit rate scalable compressed bitstream with the property that the encoder output for every bit rate is a prefix of the output for any higher bit rate. Well known examples of embedded image coders are JPEG2000 \cite{3} and SPIHT \cite{4} coders. The most relevant feature of these embedded coders is that the generated output bitstream is hierarchically organized in terms of distortion reduction and the data that apportion a greater impact on the reduction of the distortion are coded first. Considering this feature it seems natural to apply more protection to the first segments of data and less to the others. But the exact determination of the unequal error protection policy to apply raises a challenging problem. Because of the limited channel capacity, a bigger source rate diminishes the source coder generated distortion at the expenses of a minor channel coding protection which, in turns augments the decoded image distortion. The optimal ratio between source rate and channel code rate, given a limited transmission budget, has been investigated by several authors. Chande et al. \cite{1} first provided a mathematical formulation of distortion optimal unequal protection as a constrained optimization problem to be solved within the framework of dynamic programming. They consider an embedded source coder whose output stream is partitioned into fixed-length source cells and a finite set of channel block codes, each of them having a different code rate. The policy that sequentially assigns potentially different codes to different source cells (thus generating packets of different length) is determined by an exact algorithm of quadratic complexity on the total transmission rate. Their experiments show that UEP policies, generated from a given set of block codes, is superior to the EEP policy generated by any of the codes available. Banister et al. \cite{5} proposed a suboptimal method, based on Viterbi algorithm, producing a bitstream of fixed size channel packets and of quadratic complexity in the number of transmitted packets. Hamzaoui et al. \cite{2} first observed that quadratic complexity is not compatible with real time applications and addressed the problem of fast computation of distortion optimal UEP. Capitalizing on the fact that the method presented in \cite{1} provides, in linear time, source rate optimal solutions (i.e. solutions maximizing the average number of source bits received before a source-packet decoding failure) the authors provide a two step LS-method: starting from a rate optimal solution, a local search algorithm minimizes the expected distortion. Although a local minimum is not a global one, the authors claim that their method has solutions comparable to that obtained by \cite{5}, but with complexity which is linear with respect to the number of transmitted packets. Their results are tied to the fixed size channel packet scheme.

In this paper we formulate the distortion optimal UEP problem as a search problem to be solved by heuristic techniques. The distortion optimal UEP problem is reduced to the more general problem of finding a path in a graph,
where each path represents a possible protection policy and the best path is that one inducing minimal distortion. The problem is combinatorially complex and excludes a brute force approach. Heuristic information is used to evaluate a function (the expected distortion) for each node of the graph. The expected distortion is the estimated value of the minimal distortion induced by a complete path having, as prefix, the partial path from start to the given node.

This function leads to a decision about the node to expand next, selecting the code rate with the smallest expected distortion value.

The procedure is an application of the classical algorithm $A$ of Hart et al. [6], proposed in many variants depending on the way the expected distortion is evaluated. We define three evaluation functions: one with the property of being always lower than minimal distortion, the others guaranteeing a closer evaluation but only in absolute value. The more the evaluation function is closer to the minimal distortion, algorithm $A$ (in this case called $A^*$) is optimal in the solution cost; i.e. the first solution found by the algorithm is also the UEP with minimal distortion.

We give also a modified version of $A$ algorithm ensuring linear time computational complexity in the number of transmitted packets. Although suboptimal in the solution cost, this procedure improves the results given in [2].

Finally, we observe that our approach is equally suited for both cases: fixed length channel codewords (called here fixed-$N$) and variable length channel codewords with fixed length of information data (called here fixed-$K$). Most UEP schemes, with exception of [7], address only one of the two cases.

The rest of the paper is organized as follows. Section II resumes the notations we use and states the problem addressed. In Section III the algorithm $A$ is briefly described. Moreover, the heuristic functions considered to adapt the algorithm to the protection of data generated by embedded image coders are defined. In Section IV the results obtained using our approach are compared with the results obtained using the approaches presented in [1] and [2] for the N-fixed case. On the other hand the results for the K-fixed case are not reported here, since, to the best of our knowledge, there are no data nor results useful as benchmark in the literature. Finally in Section V major conclusions are drawn.

II. BASIC TERMINOLOGY

Let us consider a source-channel coding system, where the source coder is embedded and the channel codes can be chosen among a finite set of punctured turbo codes $C = \{C_1, \ldots, C_J\}$ with rates $r(C_1) < \ldots < r(C_J)$. We assume that the embedded stream is partitioned into $S$ “cells” of length $K_1, \ldots, K_S$ bits and that $\pi \equiv [C_{j(1)}, \ldots, C_{j(i)}, \ldots, C_{j(N)}], \ N \leq S$, is the protection policy that assigns to the $i^{th}$ cell the channel code $C_{j(i)}$ such that the length $L_i$ of the corresponding codeword (transmission packet) satisfies the relation $r(C_{j(i)}) \cdot L_i = K_i$. Packets are sent over a noisy channel. If a packet is decoded correctly, then the next packet is considered; otherwise the decoding is stopped. Let $p(j, \pi)$ be the probability of at least one error in the $j^{th}$-decoded packet, when the $\pi$ policy is used, and let

$$q(i, \pi) = \begin{cases} p(1, \pi) & i = 0 \\ \prod_{j=1}^{i} [1 - p(j, \pi)] p(i + 1, \pi) & 1 \leq i \leq N - 1 \\ \prod_{j=1}^{N} [1 - p(j, \pi)] & i = N \end{cases}$$

be the probability of decoding exactly the first $i$-packets correctly. We have $\sum_{i=1}^{N} q(i, \pi) = 1$.

Given $N_s$, the number of pixels in the source image, let $R_s^i = \frac{B_i}{N_s}$ the source rate of the $i^{th}$ cell expressed in bit per pixels (bpp), and let

$$R_\pi = \sum_{i=0}^{N} \frac{R_s^i}{r(C_{j(i)})}$$

the bit rate (bpp) of the policy $\pi$. Using these definitions, our problem can be formulated as a minimization problem

$$\min_{\pi} \sum_{i=0}^{N} D(i) q(i, \pi) \text{ s.t. } R_\pi \leq B,$$

where $D(i)$ is the residual distortion after successfully receiving the first $i$ packets, $B$ is the total bit rate and $g(\pi)$ is the expected distortion of the reconstructed image using the policy $\pi$.

We distinguish two different schemes:

- fixed-$N$: in which the number $L_i$ of bits of the channel codewords is fixed,
- fixed-$K$: in which the number $K_i$ of bits of the source data in each packet is fixed.

Then in the first case, the number $N$ is independent from $\pi$ and equal to $\lceil B / B_s \rceil$. On the other hand in the second case, $N$ is dependent from the protection policy $\pi$, given the constraint $R_\pi \leq B$.

III. HEURISTIC DETERMINATION OF DISTORTION

OPTIMAL UEP

Firstly, we observe that the problem can be formulated as the more general problem of finding a path in a graph.

The graph is defined by means of an initial node $s$ and a successor operator $\sigma$ whose value, for each node $n$, is the set of pairs $\sigma(n) = \{(m_j, r(C_{j})) : j = 1, \ldots, J\}$. The rate code $r(C_{j})$ is the label of the arc connecting $n$ to the successor node $m_j$ of $n$. If no merging rules are defined between the nodes, each successor node has only one parent and the graph is in fact a tree.

A path from node $n_1$ to node $n_k$ is an ordered set of nodes $[n_1, n_2, \ldots, n_{k-1}, n_k]$ where $n_{i+1}$ is a successor of $n_i$. A path $\pi$ is extended one step applying the $\sigma$ operator to the last node of the path, obtaining a set $\sigma(\pi)$ of $J$ paths.

While a generic path $\pi$ from $s$ to a generic node $n$ represents a protection policy defined by the labels of the arcs constituting the path, a generic node $n$ represents the fraction $B - R_\pi$ of bit total rate remaining available as well the total rate $\sum_{i=1}^{N(\pi)} R_i^i$
of source bits transmitted. In the fixed-N case, two nodes having the same depth (i.e. the same number of considered packets) in the tree have the same bit total rate remaining available, while, in the fixed-K case, they have the same total rate of source bits transmitted.

Since two nodes, representing the same total bit rate remaining available and the same total source bit rate transmitted, are equivalent, it is possible to merge them and then the tree becomes a direct acyclic graph. In [5], with reference to the fixed-N case, an example of such a graph is given with the number of nodes growing quadratically with the number k of the considered packets, while in a tree the number of the nodes grows exponentially (J^k if J = |C| is the number of codes that compose the set C, i.e. the branching factor of the tree). Looking at Fig. 1, the graph is defined setting r(Cj) ∈ {1, 2, 3, 4}, s = (0, 0) and σ(n, k) = ((n + nj, k + 1), r(Cj)) where nj is the numerator of the fraction representing the code rate r(Cj).

In the jargon of dynamic programming [8], the nodes having the same distance (computed as the number of arcs) from the initial node represent a partial process: the decision at one stage (the choice of the next code rate) lets the system evolve from one state into a state in the next stage.

A path π is complete if it exhausts the budget B, otherwise is partial. Complete paths individuate terminal nodes. The problem here is then to find a complete path π minimizing g(π). Since g(π) can be considered as the cost of the specific path π, the described problem can be viewed as an instance of an optimization search problem.

In the following we assume that a generic path π = [Cj1, ..., Cj(N)] satisfies the constraint r(Cj(i−1)) ≤ r(Cj(i)), i = 1, ..., N − 1. This constraint reduces the branching factor of the graph (dotted arcs in Fig. 2 are no more considered) and, as observed in [2], reduces the set of candidate paths from J^N to (J + N − 1)N.

Heuristic evaluation functions are often used to guide the search process for a minimum cost path [6].

Definition 1: An admissible heuristic evaluation function is a function f that, for a minimal cost path π with π′ = prefix(π), satisfies

\[
\begin{align*}
0 & \leq f(\pi) \leq g(\pi) \\
0 & \leq f(\pi) \leq g(\pi)
\end{align*}
\]

In our case, f should give an evaluation of the expected distortion of the minimal path having π′ = [Cj1, ..., Cj(k)], k ≤ N as given prefix.

Let we consider a search algorithm over the graph that finds the optimal distortion path after a certain number of iterations. Let we define a queue in which we store, at each iteration of the search algorithm, the partial paths over the graph, the corresponding values of the evaluation function and the corresponding last node of the given path. Since we assume, at each iteration of the search algorithm, that the most likely prefix (present in the queue) of the optimal complete distortion path is the partial path corresponding to the minimum value of the heuristic evaluation function, we can consider the following search algorithm [6]:

**Algorithm A:**
1) Let the initial queue consist of the zero length path π = [s], f(π) = D(0) and s as the corresponding last node;
2) If the queue is empty exit with failure;
3) Select the first path of the queue;
4) If the path is complete terminate the algorithm;
5) Otherwise apply the σ operator to the last node of π and compute f for each extended path obtained. Insert the extended paths in the queue, maintaining the queue \( j \)-sorted with smaller values in front. If two or more paths reach the same node, retain in
the queue only the path with minimum f-value. Go to step 2.

It is easy to verify that the algorithm terminates with a minimal distortion path if the evaluation function $f$ is admissible. By contradiction, let us suppose that the first complete path $\pi$ extracted from the queue is not the minimal one and let $\pi'$ be the first path in the queue whose completion $\tilde{\pi}$ is the minimal distortion path. In this case, however, considering Def. 1 and the construction feature of the defined queue, we have:

- $g_\pi = f_\pi$ since $\pi$ is complete (by Def. 1)
- $f_\pi \leq f_{\pi'}$ since $\pi'$ is not in the first position in the $f$-sorted queue
- $f_{\pi'} \leq g_\tilde{\pi}$ since $\pi'$ is a prefix of $\tilde{\pi}$ (by Def. 1)

this implying that $g_\pi \leq g_\tilde{\pi}$ in contradiction with the hypothesis.

As $A$ progresses in examining the graph we say that a node is expanded if we have generated its successors. It is worth noting that the number of expanded nodes during the search process is a measure of complexity for the algorithm.

As observed in [6], $A$ is not an algorithm but a family of algorithms, depending on the heuristic chosen. For this reason, to better investigate the behavior of the algorithm applied in our case, we consider different heuristics:

- H1: obtained relaxing both fixed-K and fixed-N constraint;
- H2: obtained relaxing the total rate constraint;
- H3: using the rate optimal path.

H1. Given $\pi' = [C_{j(1)}, \ldots, C_{j(k)}]$, we define the expected distortion of the minimal path having $\pi'$ as prefix, as

$$f_{\pi'} = \sum_{i=0}^{k+1} D(i) q(i, \pi)$$

where $\pi = [C_{j(1)}, \ldots, C_{j(k)}, C_{j(k+1)}]$ and

$$C_{j(k+1)} = \text{argmin}_{C \in \mathcal{C}, r(C) \geq r(C_{j(k)})} (D(k+1) q(k+1, [C_{j(1)}, \ldots, C_{j(k)}, C]))$$

assuming $B - R_{\pi}$ as the $k + 1$-packet rate.

H2. Given $\pi' = [C_{j(1)}, \ldots, C_{j(k)}]$, we define the expected distortion of the minimal path having $\pi'$ as prefix, as

$$f_{\pi'} = \sum_{i=0}^{k+M} D(i) q(i, \pi)$$

where $\pi = [C_{j(1)}, \ldots, C_{j(k)}, C_{j(k+1)}, \ldots, C_{j(k+M)}]$ with $C_{j(k+m)} = C_{j(k)}$ $m = 1, \ldots, M$ and

$$M = \begin{cases} \lfloor \frac{(B - R_{\pi}) C_{j(k)}}{R^*} \rfloor & \text{fixed-K: } R^* \text{ cell source rate} \\ \lfloor \frac{(B - R_{\pi}) C_{j(k)}}{R_k^*} \rfloor & \text{fixed-N: } R_k^* \text{ source rate of } k^{th} \text{ cell}. \end{cases}$$

It is straightforward to demonstrate (proof omitted here) that $f$ satisfies $f_{\pi'} \leq g_\pi$ and $f_\pi = g_\pi$, so that $A$, with this heuristic, is guaranteed to find a minimal distortion path.

H3. The third heuristic is directed by the rate optimal policy (RO). We recall [1] where a policy $\pi_R$ is rate optimal if maximizes the expected number of correctly decoded bits

$$\max_{\pi} \sum_{i=0}^{N} V(i) q(i, \pi) \quad \text{s.t.} \quad R_\pi \leq B,$$

where $V(i), i \geq 1$ is the number of source bits after successfully receiving the first $i$ packets ($V(0) = 0$). Given $\pi' = [C_{j(1)}, \ldots, C_{j(k)}]$ and the RO path $\pi_R$, we define the expected distortion of the minimal path having $\pi'$ as prefix, as

$$f_{\pi'} = \sum_{i=0}^{N} D(i) q(i, \pi)$$

where $\pi = [C_{j(1)}, \ldots, C_{j(k)}, C_{j(k+1)}, \ldots, C_{j(N)}]$, $N$ is the length of $\pi$ and $[C_{j(k+1)}, \ldots, C_{j(N)}]$ is the $(N - k)$-tail of $\pi_R$.

Observe that only the second heuristic is admissible, the first and the third are not. In the following we refer to the algorithms generated by the three heuristics as $A_1$, $A_2$, $A_3$.

Even if $A$ expands the fewest possible nodes necessary to find the optimal path than any other algorithm, if heuristic function $f$ is not a sufficiently good approximation of the optimal distortion path, time and space requirements may be exceedingly large. Usually $A$ runs out of space before running out of time.

Thus we implemented also a suboptimal but computationally simpler version of $A$, called Beam search. This is like $A$, but with a fixed number of queue elements, the $q$ best, considered at each iteration. Worst case Beam search complexity is linear in $qN$, assuming the search tree has depth $N$.

### IV. Results

In this section we report our results about Beam and $A$ algorithms in terms of expected distortion versus time complexity. In our experiments we assume

$$y_{JPG2}(x) = 2550.34 - 2550.89 \exp^{-0.0025x^{-0.99}}$$

$$y_{SPIHT}(x) = 1422.99 - 1424.64 \exp^{-0.0053x^{-0.99}}$$

as analytical models [9] of the operational distortion curves of standard $[512 \times 512]$ greyscale Lena image, encoded respectively by JPEG2000 and SPIHT. It is worth noting that in (1) and (2) $x = \frac{N}{N}$ is the compression ratio expressed in bit per pixel, and $y$ is the associated residual MSE distortion. Thus the expected MSE distortion, considering $k$ cells, is computed as

$$\sum_{i=0}^{k} D(i) q(i, \pi)$$

where

$$D(i) = \begin{cases} y_{JPG2}(x(i)) \\ y_{SPIHT}(x(i)) \end{cases} \quad \text{and} \quad x(i) = \sum_{j=1}^{i} R_i^*.$$
We consider data and results in [2] as benchmark. For this reason, we adopt the same conditions in terms of BER of the BSC channel, set of turbo codes considered, packet decoding error probabilities and cell length. To evaluate and compare execution times, we implemented their algorithms in our environment. The CPU time was measured in seconds on a 1.6 GHz Intel Celeron M processor 380. The programs were written in Prolog and compiled with SWI-Prolog 5.6.4.

In the first set of experiments we considered the case of Lena image compressed by the JPEG2000 encoder and protected with a concatenation of rate compatible turbo (RCPT) codes and a 16-CRC coder. Packets are transmitted over a binary symmetric channel (BSC) with crossover probability equal to $BER = 0.1$. The RCPT are generated using the (31, 27)OCT generator polynomials and bit tailing and code termination is allowed, then 4 bits are used to reset the turbo encoder into a state of all zeroes. Rates are chosen in the set $R = \{4/12, 5/12, 6/12\}$, with corresponding packet decoding error probability $p(4/12) = 0.00001$, $p(5/12) = 0.00003$ and $p(6/12) = 0.88$ respectively.

In Table I performances of $A_1$ with the three heuristic previously defined, are compared. Four target transmission rates are considered. For each rate, the search space is characterized by two quantities: tree depth, equal to the number of transmitted packets, and the number of candidate paths, equal to the number of policies satisfying the constraint of non-decreasing code rate. Performances of the three algorithms $A_1$, $A_2^*$, $A_3$ are measured by MSE distortion, CPU time and number of expanded nodes (EN) during the search process. For $A_3$ the reported CPU time includes the time spent to compute RO-path.

Being $A_2^*$ admissible, its MSE values are the minimum values of distortion attainable. Consequently, its results represent the bound on achievable MSE. However, observing Tab. I, we can note that this optimal result is obtained in exchange of a very low efficiency in terms of computational time and amount of memory required. On the other hand the heuristic of the algorithm $A_1$, although not admissible, (thus not guaranteeing to find a minimal distortion path) ensures a fast computation time while maintaining, in this specific case, the same values of distortion.

In Tab. II the performances of the distortion optimal (DO) and of the rate optimal (RO) algorithms presented in [1] and the local search (LS) algorithm presented in [2] are compared with the results obtained by the Beam search algorithm, where the heuristic of $A_1$ is adopted and the queue length is fixed equal to $q = 3$. With these parameters the Beam CPU time decreases dramatically, while maintaining the same accuracy of $A_1$ in terms of expected MSE.

In the second set of experiments we considered the image compressed by SPIHT encoder and transmitted over a BSC channel with crossover probability equal to $BER = 0.01$. The packet length is fixed equal to 2048 bits and a 32 bits-CRC code is used. The RCPT rate codes are chosen in the set $R = \{20/30, 20/29, 20/28, 20/27, 20/26, 20/25, 20/24\}$ with the corresponding decoding packet error probability: $p(20/30) = 0$, $p(20/29) = 0.0012$, $p(20/28) = 0.002$, $p(20/27) = 0.00032$, $p(20/26) = 0.0076$, $p(20/25) = 0.000354$, $p(20/24) = 0.01418$.

In this case, since the branching factor of the tree search space is large, the number of candidate paths becomes very large (for total transmission rate 1.0, seven order of magnitude larger compared with the search space of the first set of experiments). As we have explained in Section III, $A_3$, although optimal, has a computational complexity that remains exponential on the tree depth. In fact $A_1$ and $A_2^*$ consumed the system memory resources before terminating, even at rate 0.25. For this reason we do not consider these two algorithms for the comparison. On the other hand performances of Beam algorithm based on the $H_1$ heuristic and with the queue length equal to $q = 3$, remain good.

In Table III the results obtained by Beam, RO, LS and $A_3$ algorithms are compared. It is worth noting that the MSE performances of Beam, LS and $A_3$ are practically equal; LS is slightly faster than $A_3$, while Beam runs even 8 times faster than LS, as shown in Fig. 3. In this case, since $A_3$ is not admissible we need to define a MSE bound useful to evaluate performances. Under the hypothesis that the operational distortion curve $y(x)$ is convex and non increasing, and $\pi_d$, $\pi_r$ are, respectively, a distortion optimal and rate optimal policies, Hamzaoui et al. [2] proved

$$\sum_{i=0}^{N} D(i)q(i, \pi_d) \geq y(E(x(N))) \quad E(x(N)) = \sum_{i=0}^{N} x(i)q(i, \pi_r).$$

Thus the bound in Table III is computed as $y_{SPIHT}(E(x(N)))$, where $\pi_r$ is the RO-policy. Finally, in Tab. IV we show, for the case of total bit rate equal to 1.0 (for the second experiment), the protection policies generated by the considered algorithms to compare the difference between the selected optimal paths.

V. CONCLUSION

The problem of optimal distortion-based unequal error protection in progressive image transmission, using a heuristic approach is addressed. Different heuristic evaluation functions in conjunction with algorithm $A$ have been considered. To speed up the algorithm we proposed also to constraint its
search space. Numerical results showed that this sub-optimal algorithm dramatically improves A performance, in terms of space and time complexity, while maintaining the MSE values near to the MSE bound. The proposed solution outperformed the results obtained by [1], [2].

REFERENCES


TABLE I
CPU time in seconds and MSE distortion for A1, A2, A3 algorithms. Distortion curve: y_{JP2}, packet length: 4096 bits, code rates: {4/12, 5/12, 6/12}, probabilities of packet decoding error: p(4/12) = 0.00001, p(5/12) = 0.0003, p(6/12) = 0.88.

<table>
<thead>
<tr>
<th>Total rate</th>
<th>Tree depth</th>
<th>Candidate paths</th>
<th>CPU</th>
<th>MSE</th>
<th>A1</th>
<th>CPU</th>
<th>MSE</th>
<th>EN</th>
<th>A2</th>
<th>CPU</th>
<th>MSE</th>
<th>EN</th>
<th>A3</th>
<th>CPU</th>
<th>MSE</th>
<th>EN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>16</td>
<td>153</td>
<td>&lt;0.01</td>
<td>59.99</td>
<td>23</td>
<td>0.06</td>
<td>59.99</td>
<td>125</td>
<td>0.02</td>
<td>59.99</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>32</td>
<td>561</td>
<td>0.02</td>
<td>30.46</td>
<td>68</td>
<td>0.48</td>
<td>30.46</td>
<td>485</td>
<td>0.03</td>
<td>30.46</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>48</td>
<td>1225</td>
<td>0.05</td>
<td>20.46</td>
<td>190</td>
<td>1.36</td>
<td>20.46</td>
<td>904</td>
<td>0.06</td>
<td>20.46</td>
<td>48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>64</td>
<td>2145</td>
<td>0.14</td>
<td>15.41</td>
<td>391</td>
<td>4.48</td>
<td>15.41</td>
<td>1927</td>
<td>0.09</td>
<td>15.41</td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE II
CPU time in seconds and MSE distortion for Beam, Distortion Optimal, Rate Optimal and Local Search algorithms. Distortion curve, packet length, code rates and probabilities of packet decoding error identical to that of Table I

<table>
<thead>
<tr>
<th>Total rate</th>
<th>Tree depth</th>
<th>Candidate paths</th>
<th>Beam</th>
<th>CPU</th>
<th>MSE</th>
<th>DO</th>
<th>CPU</th>
<th>MSE</th>
<th>RO</th>
<th>CPU</th>
<th>MSE</th>
<th>LS</th>
<th>CPU</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>16</td>
<td>153</td>
<td>&lt;0.01</td>
<td>59.99</td>
<td>0.03</td>
<td>59.99</td>
<td>&lt;0.01</td>
<td>59.99</td>
<td>&lt;0.01</td>
<td>59.99</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>32</td>
<td>561</td>
<td>0.02</td>
<td>30.46</td>
<td>0.09</td>
<td>30.46</td>
<td>0.02</td>
<td>30.97</td>
<td>0.02</td>
<td>30.46</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>48</td>
<td>1225</td>
<td>0.02</td>
<td>20.46</td>
<td>0.27</td>
<td>20.46</td>
<td>0.03</td>
<td>21.21</td>
<td>0.05</td>
<td>20.46</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>64</td>
<td>2145</td>
<td>0.02</td>
<td>15.41</td>
<td>0.48</td>
<td>15.41</td>
<td>0.05</td>
<td>16.33</td>
<td>0.05</td>
<td>15.41</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE III
CPU time in seconds and MSE distortion for Beam, Rate Optimal, Local Search and A3 algorithms. Distortion curve: y_{SPHT}, packet length: 2048 bits, code rates: {20/30, 20/29, 20/28, 20/27, 20/26, 20/25, 20/24}, probabilities of packet decoding error: p(20/30) = 0, p(20/29) = 0.0012, p(20/28) = 0.0002, p(20/27) = 0.00032, p(20/26) = 0.00076, p(20/25) = 0.00054, p(20/24) = 0.01418.

<table>
<thead>
<tr>
<th>Total rate</th>
<th>Tree depth</th>
<th>Candidate paths</th>
<th>Beam</th>
<th>CPU</th>
<th>MSE</th>
<th>A3</th>
<th>CPU</th>
<th>MSE</th>
<th>RO</th>
<th>CPU</th>
<th>MSE</th>
<th>LS</th>
<th>CPU</th>
<th>MSE</th>
<th>Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>32</td>
<td>2760681</td>
<td>0.02</td>
<td>32.38</td>
<td>0.08</td>
<td>32.37</td>
<td>0.03</td>
<td>33.87</td>
<td>0.05</td>
<td>32.37</td>
<td>31.39</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>64</td>
<td>131115085</td>
<td>0.05</td>
<td>17.03</td>
<td>0.23</td>
<td>17.03</td>
<td>0.08</td>
<td>19.28</td>
<td>0.20</td>
<td>17.03</td>
<td>16.43</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>96</td>
<td>1346548665</td>
<td>0.06</td>
<td>11.50</td>
<td>0.50</td>
<td>11.49</td>
<td>0.17</td>
<td>12.84</td>
<td>0.50</td>
<td>11.50</td>
<td>11.08</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>128</td>
<td>7177979809</td>
<td>0.09</td>
<td>8.59</td>
<td>0.95</td>
<td>8.59</td>
<td>0.34</td>
<td>9.83</td>
<td>0.88</td>
<td>8.59</td>
<td>8.27</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE IV
UEP generated policies for Beam, Rate Optimal, Local Search and A3 algorithms when the total rate is equal to 1.0 and the crossover probability of the BSC channel is equal to BER = 0.01.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Code Rates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beam</td>
<td>20/30 20/29 20/28</td>
</tr>
<tr>
<td>RO</td>
<td>0 0 0</td>
</tr>
<tr>
<td>LS</td>
<td>0 0 8</td>
</tr>
<tr>
<td>A3</td>
<td>0 0 2</td>
</tr>
</tbody>
</table>