Avaliação de Códigos Fontanais

Aluno: Andréa Micheli Alzuguir

Orientador: Weiler A. Finamore

Introdução

A comunicação se realiza através da transmissão da mensagem que desejamos enviar, através de um meio, e esperamos que o receptor consiga determiná-la de maneira correta mesmo diante de adversidades.

Em um sistema de comunicação real depara-se com diferentes problemas, em especial, certas perturbações introduzidas pelo meio de comunicação, geram apagamento de informações. O que ocorre quando se envia informação através de um meio modelado como um canal com apagamento, é que a informação recebida pode ser, reconhecidamente, diferente da que foi transmitida. No nosso trabalho estamos interessados em sistemas em que a informação produzida pela fonte é binária e consideramos que estas informações binárias são agrupadas em blocos (ou **pacote de mensagem**) formando a mensagem a ser enviada. Nessas condições, portanto, pode ocorrer que a mensagem produzida pela fonte (uma seqüência de pacotes) é enviada através do canal e pode ser recebida com alguns pacotes apagados (irreconhecíveis). A internet é um exemplo simples e clássico onde a comunicação é feita através da troca de informação sob forma de pacotes.

Uma solução usada para o problema do recebimento de informações apagadas (pacotes reconhecidamente com erros) usa um canal de retorno (feedback channel), entre o receptor e o transmissor, através do qual faz-se uma solicitação de retransmissão dos pacotes corrompidos (ou apagados). No entanto, um método recentemente descoberto, propõe como solução proteger os pacotes, antes de enviá-los através do canal, submetendo-os, a um processamento denominado por códigos fontanais (trabalharemos com a implementação conhecida como códigos LT - Luby Transform).

Os códigos fontanais foram desenvolvidos de maneira a gerar, na saída do codificador, uma seqüência de bits de tamanho muito grande (potencialmente contendo um número infinito de bits), ditos **pacotes codificados**, gerados a partir dos pacotes de entrada, que agregam a informação original (pacotes gerados pela fonte) e informação redundante. Os pacotes codificados são produzidos através de operações simples. Usando os pacotes codificados que foram recebidos corretamente (não foram apagados) pode-se construir um receptor que é capaz de recuperar a informação original. Podemos visualizar tal processo através do diagrama em blocos ilustrado abaixo.

Figura 1: Sistema de comunicações

O código fontanal usando no codificador e o decodificador ilustrado no diagrama da Figura 1, foi construído usando uma distribuição de grau conhecida como Sóliton Robusta Melhorada. A distribuição Sóliton Robusta foi utilizada para alguns testes iniciais e entendimento de aspectos teóricos destes códigos. Alguns aspectos relacionados às ferramentas usadas no desenvolvimento da pesquisa serão definidos nas seções seguintes.

Desta maneira, apresentaremos aqui o estudo que foi realizado sobre o sistema de codificação-decodificação fontanal, para canais com apagamento, ou seja, códigos LT. E apresentaremos também a análise de desempenho que foi realizada, mostrando que o uso de códigos LT pode representar uma vantagem para aumentar a confiabilidade da comunicação.

Canal

Para compreendermos melhor os códigos LT é preciso um estudo e entendimento mais detalhado dos modelos de canais com apagamento, ou seja, através do qual a mensagem é enviada e, na saída, recebida corrompida (ditas apagadas).

Na pesquisa desenvolvida foi utilizado um canal binário com apagamento chamado de BEC (*Binary Erasure Channel*). Este modelo é bastante utilizado na Teoria da Informação e Códigos pois é um dos canais que modela de forma útil alguns sistemas práticos.

Neste canal apenas os símbolos “0” ou “1” (*bits*) podem ser transmitidos como informação --- isto é, são aceitos em sua entrada. Como não se trata de um canal ideal, podemos receber os *bits* enviados (sem erro) ou podemos ter, na saída do canal um apagamento, ou seja, um símbolo que não sabemos se é o *bit* “0” ou o *bit* “1”. Vale ressaltar também que o BEC não é um canal que introduz erro, pois sempre que o *bit* “0” ou o *bit* “1” é recebido, temos certeza que estes foram, de fato, os *bits* enviados --- isto quer dizer que ao receber um o *bit* “0” ou o *bit* “1”, a confiabilidade de estar correto é 100%. A Fig. 2 a seguir ilustra um sistema onde a informação transmitida é uma seqüência de zeros e uns e a seqüência recebida é uma seqüência de zeros, uns ou apagamentos. Se representarmos o símbolo correspondente ao apagamento por “?” podemos representar o canal com apagamento pelo diagrama da Fig. 3.

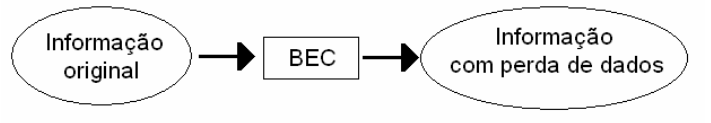

Figura2: Diagrama do sistema de comunicação com BEC

Se, por exemplo, o pacote de 12 símbolos binários “001 101 100 111” é enviado através do BEC é possível que o pacote recebido seja “001 ?01 10? 111”. Neste caso dois símbolos binários fora apagados ao atravessar o canal.

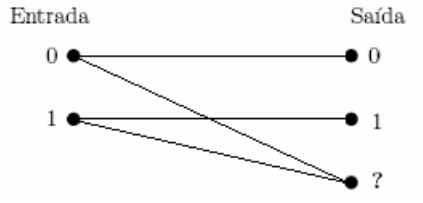

Figura 3: Representação do Canal BEC

O BEC fica completamente caracterizado quando conhecemos a probabilidade de ocorrência de um apagamento --- se denominarmos por x o símbolo de entrada do canal e por y o símbolo de saída dizemos que

$$Rho = P(y=?|\ x=0) = P(y=?|\ x=1)$$

é a probabilidade de apagamento do canal.

O sistema codificador-decodificador está inteiramente relacionado com a probabilidade de apagamento do canal, pois é esta que definirá a redundância que deverá ser introduzida na codificação para que seja realizada uma decodificação sem erros. Assim, algumas simulações foram realizadas e chegou-se a seguinte conclusão: quanto maior a probabilidade de apagamento, maior é a quantidade de símbolos codificados que devemser transmitidos, ou seja, estas duas quantidades são diretamente proporcionais. No gráfico da Fig. 4 apresentamos a capacidade do canal BEC (C=1-rho) assim como a capacidade de dois outros canais (BSC e Z) derivados do BEC. Nele observamos também que os canais BSC e Z possuem uma capacidade

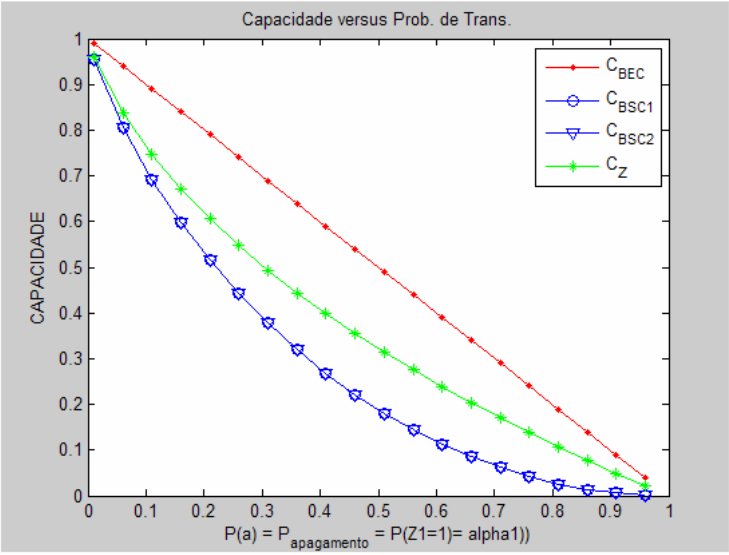

Figura 4: Capacidade do Canal X Probabilidade de Transmissão

inferior ao BEC, uma vez que estes canais foram obtidos transformando-se o BEC ()um canal com saída ternária) em um canal com saída binária. O BEC é transformado no canal Z substituindo-se os apagamentos pele]o *bit* "1" (i.e., introduzindo erro se o *bit* enviado foi o *bit* "0". Da mesma forma o canal BEC foi pode ser transformado em um BSC (substituindo-se aleatoriamente os apagamentos do BEC por *bits* "0" ou "1"). O canal Z e o BSC são canais que introduzem erro.

No gráfico da Fig. 5 a probabilidade de erro versus qualidade-do-canal, ao se transmitir *bits* através do BEC e do BSC são apresentadas.

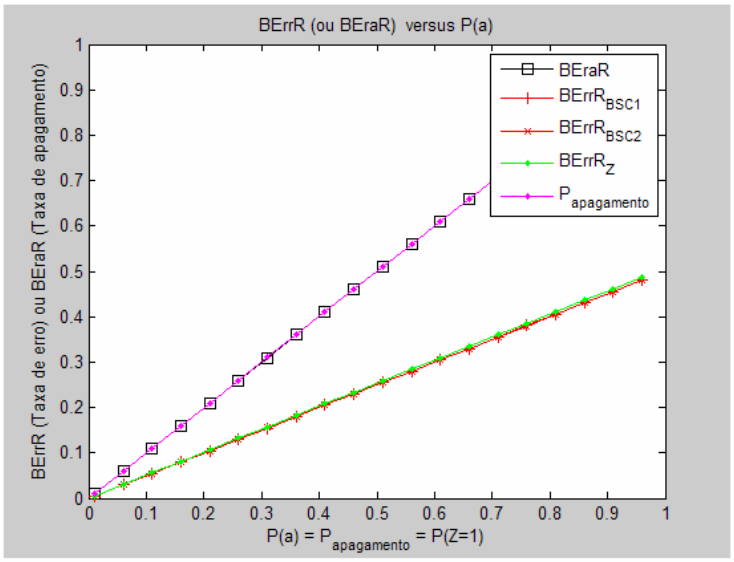

Figura 5: Taxa de apagamento versus Apagamento (ou erro).

A Fig. 6, a seguir, mostra o resultado dos testes realizados quando os bits que compõem uma imagem (neste caso a imagem, em formato TIF, conhecida por LENA.TIF), sem a proteção de um código são enviados através do canal. A imagem original (produzida pela fonte de informação) é exibida no lado esquerdo da Fig. 6. A probabilidade de erro do canal (rho) foi variada desde 0.01 ao valor 0.99, em incrementos de 0.05. Ao final, com rho =0.99 (um canal extremamente hostil) obtemos, na saída, a imagem apresentada no lado esquerdo da Fig.6..

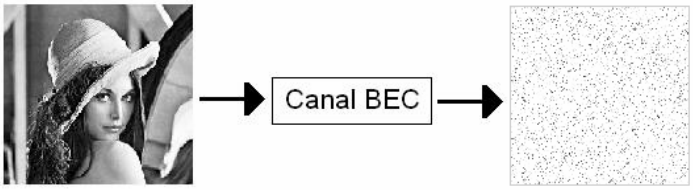

Figura 6: Lena_entrada → BEC → Lena_saída

Codificador

O processo de codificação LT é considerado universal dado que sua eficiência aumenta quanto maior o tamanho do bloco de informação que se está trabalhando, além

de, por trabalhar com operações do tipo XOR, os codificadores executam a codificação de maneira rápida.

O processo convive com o seguinte dilema: considera-se como entrada do codificador um bloco de tamanho k, ou seja, a informação é formada por k símbolos binários; sua saída é um novo bloco com tamanho n >= k que contem a informação inicial codificada. Dessa forma, a taxa de transmissão do codificador (R) e a redundância (r) são dadas, respectivamente, por:

R= k/n
r=n-k

Essa redundância é inserida visando reduzir os efeitos da perturbação introduzida pelo canal sobre a informação recebida --- como foi mostrado anteriormente, em canal com probabilidade de apagamento 0,99 a informação recuperada (que corresponde à informação observada na saída do canal) muita pouca relação tem com a mensagem que foi enviada (imagem LENA). Assim, ao codificarmos a informação original, antes de envia-la através do canal, aumentamos o tamanho do vetor inicial para que caso haja alguma perda de informação consigamos, usando a redundância, recuperar corretamente cada *bit* enviado --- isto é de forma que, ainda que alguns símbolos não tenham sido recebidos corretamente, foram apagados, a mensagem original possa ser recebida sem erros ou com um pequeno número de bits errados (quando a fonte transmite imagens, a comunicação é aceitável se o número de bits errados é pequeno).

A codificação consiste, portanto, em operações de adição módulo 2 (ou XOR) realizada entre os elementos de uma cópia da informação inicial que chamaremos de **s**, onde o tamanho de **s** é |**s**|=k. A cada símbolo binário c_i, ou *bit*, do bloco de saída **c**, de tamanho |C| = n é atribuído um grau g_i. Para obter o valor de cada *bit* de saída c_i (c_i=0 ou c_i=1) g_i números i1, i2, ig , inteiros, distintos, com valores entre 1 e k são selecionados aleatoriamente e o valor de c é dado pela soma (soma módulo 2)

c_i = s_i1 + s_i2 + ... + s_ig

Os graus g_1, g_2, ..., g_n podem ser representados por um vetor **g** =(g_1, g_2, ..., g_n) de tamanho |**g**|=n que pode ser gerado de diferentes maneiras. Trabalharemos aqui com a geração aleatória que segue a distribuição de graus chamada de Sóliton Robusta (para blocos com tamanho muito pequeno, menor do que 32 bits), e Sóliton Robusta Melhorada para blocos maiores.

Em nossa implementação decidimos testar uma escolha de símbolos de entrada para realizar a soma módulo-2 que produz o símbolo c_i, determinística. Ou seja, para gerar o i-ésimo símbolo i, do vetor de saída **c**, realizamos o XOR dos g_1 primeiros símbolos de **c** (para gerar o símbolo c_1), dos g_2 símbolos seguintes de **c** (para gerar o símbolo c_2) e assim por diante. Acreditamos que esta maneira pode simplificar, sem prejuízos para o desempenho, a implementação do codificador.

A visualização desse sistema torna-se mais simples se o representarmos por um grafo bi-particionado. Podemos associar os k *bits* bloco de entrada **s** com **nós de entrada** do grafo, e os n *bits* do bloco **c**, de saída do codificador, com **nós de saída** do grafo. Os nós de entrada na posição (i_1+1), ...,(i_1+g_i), são adicionados módulo-2 e

o resultado [s_(i1+1) + s_(i1+2) + ... + s_(i1+g_i)], que constituem o símbolo de saída c_i são associados ao nó de saída c_i --- o grafo é completado unidos os nós de entrada ao nó de saída por arestas (que poderiam ser representadas por {s_(i1+1)→ c_i}, {s_(i1+2))→ c_i}, ..., {s_(i1+g_i))→ c_i}. Vejamos um exemplo.

Exemplo1:

Considere o seguinte vetor de entrada **s** = (s1, s2, s3, s4) = (1,1,0,1), uma taxa de transmissão do codificador R = 4/5 e o seguinte vetor de graus **g** = (g1, g2, g3, g4, g5) = (4, 2, 3, 1, 2). Esses são os elementos necessários para que a codificação seja realizada. O processo de codificação está representado na tabela abaixo:

Saída (**c**)	Grau (**g**)	Vizinhos (**s**)	Valores das entradas	Valor da saída
c1	4	s1 s2 s3 s4	1 1 0 1	1
c2	2	s1 s2	1 1	0
c3	3	s3 s4 s1	0 1 1	0
c4	1	s2	1	1
c5	2	s3 s4	0 1	1

Tabela 1: Processo de codificação.

A partir dessa tabela podemos construir um grafo ilustrativo.

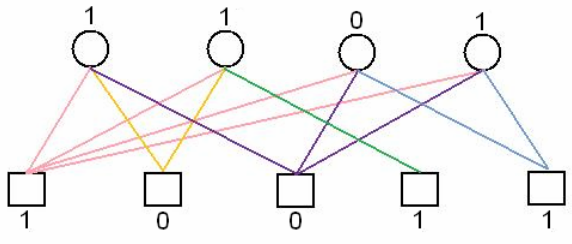

Figura 7: Grafo da codificação

Visto isso, apresentamos um resultado mais fácil de ser visualizado de nossas simulações, ou seja, no lugar de uma pequena seqüência de *bits*, como entrada, utilizaremos a imagem LENA.TIF. Trata-se, portanto, de um bloco maior e que precisa ser subdividido em uma seqüência muito grande de pacotes (cada linha da imagem, com 256 bytes, foi tomada como um pacote) --- esta seqüência de pacotes foi processada pelo codificador.

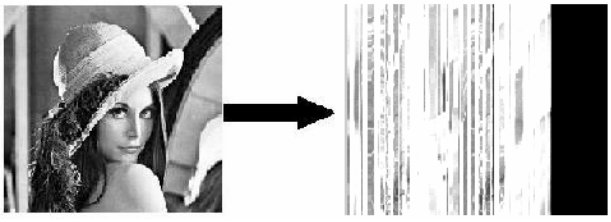

Figura 8: Lena → Lena codificada

Decodificador

É possível que o bloco **y** recebido pelo receptor seja diferente do bloco **c** produzido pelo codificador, mas vamos considerar inicialmente, para explicar o processo de decodificação que **y**=**c** (estamos supondo que o canal não apagou nenhum símbolo transmitido). Precisamos agora decodificar a informação recebida **y** e a partir desta, recuperar a mensagem original. Como houve apagamentos, o usuário poderá receber uma mensagem **u** igual à mensagem original **s** original, sem erros, ou seja, **u=s**.

Para que o processo de decodificação seja realizado é preciso que o decodificador conheça a distribuição de graus utilizada para se obter os graus iniciais, ou seja, o vetor **g**, sua entrada será o vetor **y**=**c** que é a mensagem codificada após atravessar o canal e a saída (do decodificador), que será o queremos descobrir, é o vetor **u**. Basta que o decodificador conheça o grafo mostrado anteriormente na Figura 8 (o mesmo grafo usado pelo codificador).

Para ficar mais claro, opta-se por descrever o processo empregando-se novamente o grafo. O algoritmo opera da seguinte maneira:

ALGORITMO DE DECODIFICAÇÃO

1. Primeiramente devemos encontrar um nó de entrada do decodificador (são n os nós de entrada do decodificador), ou seja, a componente y_i de **y** (entrada do decodificador), que possua grau unitário e transferimos este valor (y_i) para a saída do decodificador u_j (conectada ao nó y_j), fazendo u_j= y_i.
2. Caso não haja nenhum nó com essa característica o decodificador declara que houve **falha** no processo. Nesta situação, é preciso que mais símbolos codificados sejam enviados para que o processo de decodificação prossiga (caso isto não seja possível – re-envio - o processo termina com falha e apenas alguns bits serão recuperados).
3. Caso contrário, o processo continua de modo que todos os nós de entrada vizinhos do nó u_j (sejam estes designados por y_j1, y_j2, ..., y_jg --- estes são os nós de entrada ligados por uma aresta ao nó de saída do decodificador, u_j) têm os seus valores alterados ou atualizados, para (y_j1+u_j), (y_j2+u_j), ..., (y_jg+u_j) i.e., são adicionados, módulo-2 (XOR), a U_j.
4. O valor u_j é considerado descoberto e as arestas (y_j1→u_j), (y_j2→u_j), ..., (y_jg→u_j) são removidas, juntamente com o nó u_j, do grafo.
5. O processo se repete, utilizando agora o novo grafo (sem as arestas removidas e nó descoberto) retornado-se ao PASSO 1 até que haja uma falha ou toda a mensagem tenha sido decodificada (os valores de **u** descobertos).

O processo pode ser melhor observado e compreendido acompanhando-se o exemplo que segue.

Exemplo2:

Considere que **y** = (u_1, u_2, u_3, u_4, u_5, u_6) = (100111) é o vetor recebido. A aplicação do Algoritmo de Decodificação pode ser observada na Figura 9, acompanhando a evolução do grafo associado a cada passo do algoritmo. A seqüência de grafos mostrada na Figura 9, ilustra o processo de decodificação --- observe que neste caso a mensagem original **s** = (10011) foi recuperada pelo decodificador (ou seja **u=s)**.

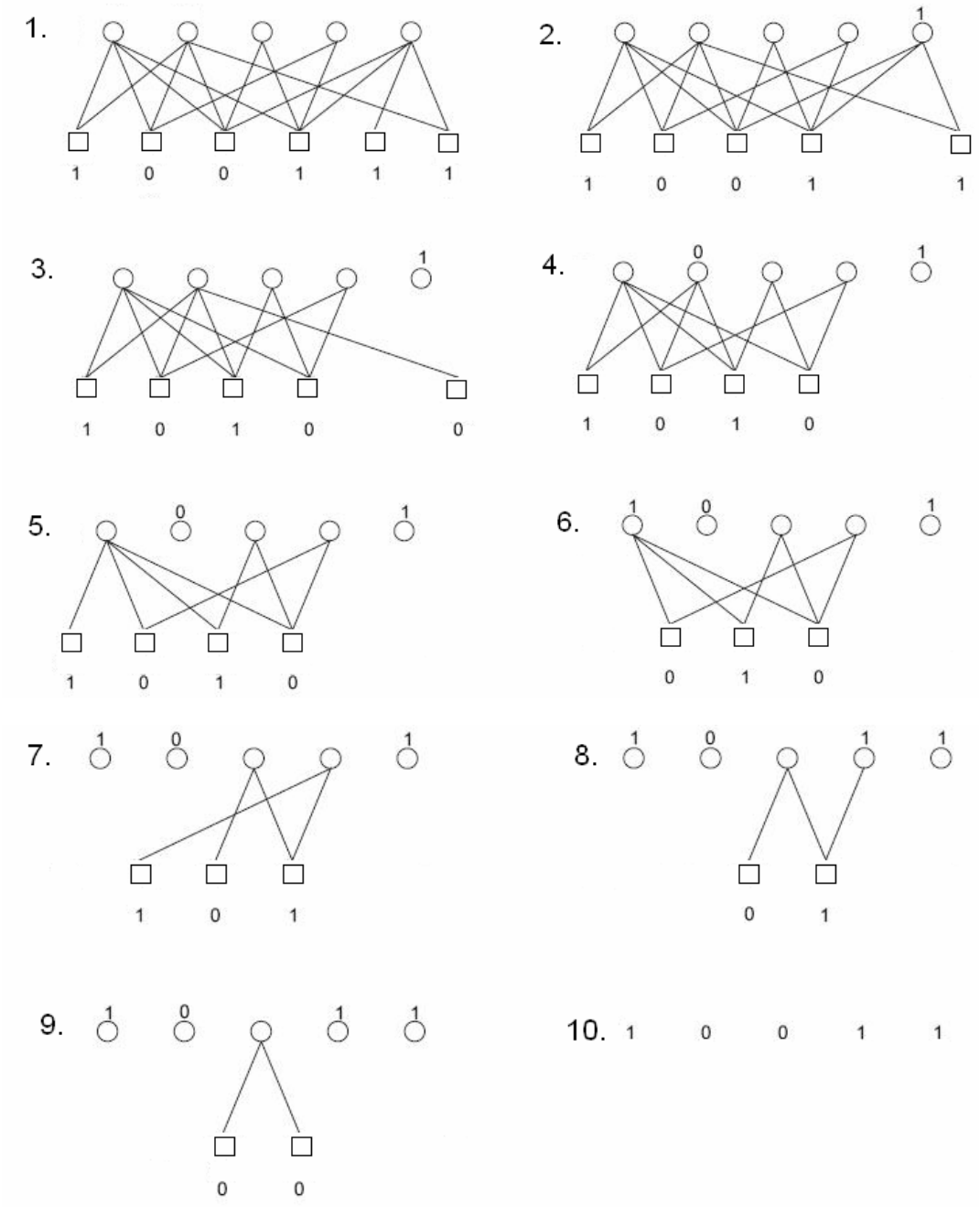

Figura 9: Grafos do processo de decodificação

Ao aplicar esta técnica de codificação-decodificação em nossa pesquisa, onde se considera que o canal introduz apagamentos observa-se dois resultados: em um deles o

processo não falha e a informação enviada é recuperada sem erro (como na Figura 9) e, outro, em que há falha de decodificação e a informação recuperada conterá apagamentos. Por exemplo, se o vetor recebido fosse **y**= (u_1, u_2, u_3, u_4, u_5, u_6) = {100?11}, é fácil verificar que o vetor à saída do decodificador seria **u** = (10?11) diferente de **s**. Neste caso, ao usuário, como os símbolos entregues devem ser "zeros" ou "uns", é necessário substituir o símbolo "?" por "0" ou "1" --- sendo esta substituição feita aleatoriamente, a informação entregue ao usuário seria, em 50% das vezes, **s**_hat = (10111)**, CONTENDO ERRO,** ou **s**_hat = (10011) sem erros outras 50% das vezes. A Figura 10 ilustra a situação onde a decodificação falhou.

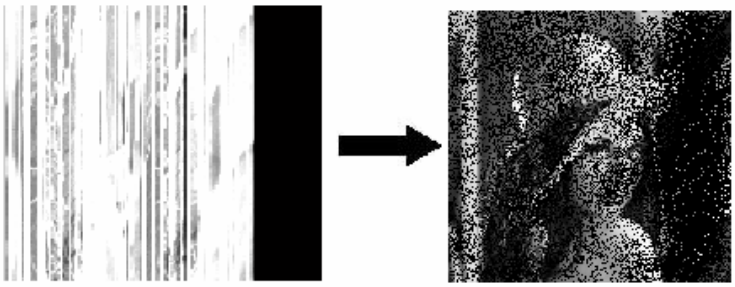

Figura 10: Lena Codificada (imagem da esquerda) e Lena Decodificada com erros, quando há falha de decodificação (imagem da direita)

Diferentes tipos de falhas podem ocorrer, se a falha ocorre logo no início do processo de decodificação, muitos símbolos, não serão descobertos (permanecerão apagados) e não se consegue, às vezes, recuperar uma mensagem que possa ser identificada como a mensagem original, no entanto se a falha ocorrer na parte final do processo a mensagem pode ser parcialmente compreendida (i.e., pode guardar uma grande semelhança com a mensagem original).[1] Além disso, como a geração dos graus é aleatória não sabemos previamente em qual parte do processo existe maior possibilidade de ocorrer esses erros, por isso, quando se deseja proteger a informação mais fortemente, adiciona-se muita redundância (i.e., trabalha-se com uma taxa de codificação R, mais baixa).

Desempenho

Através de simulações pode-se avaliar o desempenho desta técnica de codificação-decodificação --- para isto, utilizam-se diferentes parâmetros, ou seja, variados tamanhos para a imagem além de variadas taxas de transmissão do codificador (R) e redundância (r) e observa-se que parâmetros conduzem o processos a decodificação com sucesso ou satisfatória.

Em canais com apagamento, o desempenho também se mostrou eficiente. Apesar da informação passar por um meio adverso o processo de codificação e decodificação é bastante eficaz e observou-se que R deve ser inversamente proporcional à taxa de apagamento do canal, mostrando assim que é possível obter decodificação com sucesso usando um *overhead* pequeno (conforme verificado e analisado anteriormente).

[1] É importante observar, em se tratando de mensagens como imagens ou vídeos, que ter uma mensagem recuperada semelhante à mensagem original pode ser útil mas, no caso de outras mensagens (por exemplo dados bancários ou dados, genericamente) erros não são admitidos.

O gráfico a seguir, Figura 11, ilustra os conceitos aqui discutidos.

Conclusão

Neste trabalho buscou-se não apenas fazer um estudo códigos fontanais (códigos LT), como também analisar (via simulação) o desempenho de um sistema que utiliza este tipo de código.

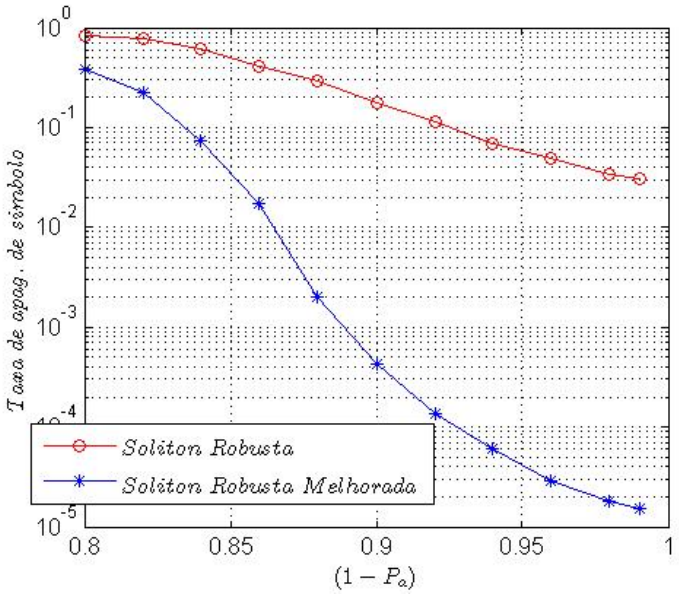

Figura 11: Taxa de apagamento versus qualidade do canal (para um overhead fixo).

O objetivo foi desenvolver as ferramentas de software (em MATLAB) para realizar a análise e observar na prática (via simulação) os resultados prometidos pela teoria. Nos programas utilizados nas simulações, implementou-se o canal BEC e segundo suas características e testamos nele os processos de codificação e decodificação LT.

O software continua em desenvolvimento. Alguns resultados (como o da Figura 11) foram obtidos com um programa desenvolvido para uma tese de mestrado[2]. O plano é prosseguir e analisamos, portanto, o desempenho para diferentes taxas de apagamento do canal (rho), partindo de um canal ideal, onde esta taxa é igual a zero, e assim aumentando-a progressivamente, analisar o desempenho em função de R e rho.

[2] Sanchez Paiba, Franklin Antonio, **Códigos Fontanais Bidimensionais para Canais com Apagamento**, Dissertação de Mestrado, PUC-Rio, 2008.

Como dito anteriormente, o resultado mais importante observado foi o de que o desempenho do sistema melhora quando o tamanho do vetor de informação aumenta. Acreditamos que esses códigos serão muito empregados nos sistemas de “telecomunicações” mais atuais.