COMPACTAÇÃO DE DADOS SÍSMICOS COM PERDA CONTROLADA

Antonyonne Soares Bessa
Universidade do Estado do Rio de Janeiro - IME/DICC - RJ
20550-013 - Av. São Francisco Xavier, 524, Maracanã, Rio de Janeiro, RJ
e-mail: tombessa@gmail.com

David Sotelo Pinheiro da Silva
Petróleo Brasileiro S.A.-Petrobrás
20031-912 - Av. República do Chile, 65, Centro, Rio de Janeiro, RJ
e-mail: david@petrobras.com

Paulo Eustáquio Duarte Pinto
Universidade do Estado do Rio de Janeiro - IME/DICC - RJ
20550-013 - Av. São Francisco Xavier, 524, Maracanã, Rio de Janeiro, RJ
e-mail: pauloedp@ime.uerj.br

Pedro Henrique Ribeiro da Silva
Universidade do Estado do Rio de Janeiro - IME/DICC - RJ
20550-013 - Av. São Francisco Xavier, 524, Maracanã, Rio de Janeiro, RJ
e-mail: pedrocp2@gmail.com

RESUMO

Levantamentos sísmicos são uma das principais ferramentas utlilizadas pela indústria de petróleo na busca de depósitos de hidrocarbonetos, resultando em arquivos com tamanhos na ordem de terabytes. Métodos de compressão sem perda não se aplicam bem a dados sísmicos, atingindo um fator de compressão de cerca de 50% do tamanho original do dado. Métodos de compressão com perda geram melhores fatores de compressão mas geralmente destroem informações importantes, relacionadas a descontinuidades. Seguindo tal motivação, o presente trabalho apresenta um novo método de compressão com perda controlada. Dado um parâmetro p, considera-se que nenhum elemento possa ter seu valor recodificado para um valor que difira mais que p vezes desse valor. Resultados experimentais em dados sintéticos atestam que o método proposto apresenta um ganho significativo em relação a métodos de compressão sem perda, atingindo um fator de compressão de cerca de 18% do dado original, mesmo quando considerado um erro máximo de 1%.

ABSTRACT

Seismic surveys are a major tool used by oil industry on finding hydrocarbon deposits, resulting in file sizes of the orders of terabytes. Lossless data compression methods do not fit well on seismic data, achieving a compression ratio threshold of about 50% of the original data. Lossy data compression methods, on the other hand, generate better compression factors but usually destroy some extremely important information on seismic data, related to discontinuities. With this motivation, this work introduces a new compression method of controlled loss. Given a parameter p, one must consider that, after applying a lossy compression method, no element have its recoded to a value wich differs from it more than p times. Experimental results on synthetic data attest that the proposed method culminates on a substantial improvement over lossless ones, achieving a compression ratio of about 18% of the original data, even considering an error factor of 1%.

KEYWORDS. Seismic Data. Data Compression. Algorithm.

Main area (P&G - O. R. in Oil and Gas Applications)

1. Introdução

Dados sísmicos são dados procedentes de informações registradas em superfícies terrestres ou marítimas, utilizados na caracterização de propriedades físico-químicas do meio de propagação de ondas acústicas, geradas por mecanismos naturais ou artificiais. Na indústria de petróleo, em particular, geofísicos e geólogos de exploração utilizam tais informações para **identificar potenciais depósitos de hidrocarbonetos [Sheriff (1995)]**. O procedimento envolvendo a obtenção de tais informações é conhecido como **levantamento sísmico**. Uma característica comum em levantamentos sísmicos é o **grande volume de dados resultante** de sua aplicação. Frequentemente, tais volumes são da ordem de **centenas de gigabytes,** com tendência atual de massas de dados resultantes na ordem de **terabytes.**

Motivados em fornecer mecanismos que facilitem a manipulação dessas grandes massas de dados, inúmeros métodos de **compactação de dados sísmicos** têm sido propostos e experimentados nos últimos anos **[Anjos (2007)][Averbuch (2001)][Villassenorl (1996)][Xie (2009)]**. Dentre os benefícios resultantes da aplicação desses métodos de compactação destacamos: a **redução do tempo de transmissão de dados** entre o local de aquisição e os centros de processamento sísmico, a **diminuição no espaço** necessário para armazenamento de dados sísmicos em disco, e o **descongestionamento do tráfego** de rede resultante da paralelização de algoritmos de processamento sísmico em ambientes computacionais com múltiplos processadores **[Panetta (2009)]**.

Em sua grande maioria, tais métodos de compactação encontram-se fortemente baseados em **transformadas matemáticas** que, em um passo inicial, transformam os dados originais para um conjunto mais bem comportado, com **maior redundância de informação.** Em seguida aplicam um método de compactação sem perdas, tais como **Huffman [Huffman (1951)]** ou **PPM [Salomon (2007)]**. Ocorre que, nesses processos, normalmente há **perda de dados,** o que constitui um problema para alguns tipos de processamentos. O presente trabalho foi inspirado em um desses métodos, desenvolvido por Anjos **[Anjos (2007)].** Ali, os dados sísmicos são previamente tratados, gerando, para cada dado sísmico, um representante. A idéia é que haja muito menos representantes que dados originais. A propriedade do conjunto de representantes, nesse trabalho, é a de **minimizar o erro médio**, na medida em que os dados originais são substituídos pelos seus respectivos representantes. A minimização do erro médio pode introduzir distorções indesejáveis, por **mascarar certos "picos" de dados**. No método proposto neste artigo, faz-se algo parecido, porém garante-se que **o erro é controlado** e estará sempre dentro de determinado parâmetro fornecido, eliminando o incoveniente apontado.

Na **Seção 2** descrevemos o método proposto. Na **Seção 3** são apresentados e discutidos os resultados computacionais obtidos a partir de dados simulados e, na **última Seção** são discutidas algumas perspectivas sobre a continuação do trabalho.

2. O método proposto

Inicialmente daremos uma **idéia geral** do método proposto, que pode ser dividido em duas grandes fases: **reorganização de dados** e **compactação**. Embora a **descompactação dos dados** não faça parte do método descrito, vamos também apresentar resultados experimentais para esse processo

Antes da compactação ser efetivada, o dado passa por uma **reorganização**, composta de dois processos. O primeiro é a **ordenação do arquivo**. O segundo é a **recodificação dos dados** que consiste na **substituição** de cada dado original **por um representante**. Cada representante substitui um intervalo de valores originais, baseado no valor da **perda aceitável**.

Após a fase de reorganização dos dados, é efetivada a **compactação**. Foi utilizada a **codificação de Huffman** baseando-se na estatística feita sobre os representantes, durante a fase de reorganização. Esta fase consiste na criação da **árvore de Huffman,** na **substituição** de cada dado original por seu representante e na **codificação** de cada dado usando o código criado.

A seguir detalhamos as etapas do processo.

2.1 Geração dos Números

Os dados sobre os quais o método vai operar são dados numéricos **simulados** e representados em ponto flutuante utilizando 32 bits. Por questões de confidencialidade não se pôde trabalhar com dados reais, mas a empresa envolvida neste trabalho forneceu uma **distribuição típica** de algumas amostras de dados sísmicos reais. A Tabela 2.1 apresenta a distribuição utilizada neste trabalho. Os experimentos envolveram amostras cujo volume variou na faixa de 100 MB a 2 GB.

Faixa Numérica	Probabilidade
[00.00000, 04.99999]	65%
[05.00000, 19.99999]	20%
[20.00000, 49.99999]	10%
[50.00000, 99.99999]	05%

Tabela 2.1 – Distribuição de dados sísmicos típicos

2.2 Ordenação dos dados

O **volume** de dados, como foi dito, **é muito grande,** em geral. Então deve-se supor que os dados possam **não caber na memória** do computador disponível. Torna-se necessário o uso da técnica geral de **ordenação externa de arquivos**, que consiste, numa primeira etapa, em particionar o arquivo original em **blocos ordenados** e, numa segunda etapa, fazer a operação de **união desses blocos [Cormen (2009)][Vitter (2008)]**. Para a criação dos blocos, utilizou-se a **Seleção por Substituição** que, por meio de um um heap, lê o arquivo, ao mesmo tempo que dá saída nos blocos ordenados. Em geral este método possibilita que sejam criados blocos com o **dobro do tamanho do heap** utilizado. Nos casos em que os dados cabem totalmente na memória, este método funciona aproximadamente como o **Heapsort [Cormen (2009)]** e é gerado apenas um bloco ordenado, não havendo necessidade da etapa de união de blocos. Para a **união dos blocos** ordenados, utilizou-se a **Intercalação Balanceada [Vitter (2008)]**, que trabalha com quatro arquivos temporários onde, em cada passo da união de blocos, dois dos arquivos são de entrada e dois de saída. No passo seguinte, essa função é invertida.

2.3 Recodificação dos Dados

A recodificação dos dados executa dois processos básicos de forma simultânea. O primeiro é a **definição de representantes**, com a criação de um **dicionário** onde se guarda cada representante. O segundo, é a geração de uma **estatística** da ocorrência dos representantes, que servirá como entrada para a criação da **árvore de Huffman** para a compactação. Esta estatística é anexada ao dicionário criado.

Um **representante** para um valor de entrada é um número cuja diferença para esse dado está dentro da perda adotada no método. Em termos matemáticos, se x é representante de y, e a perda adotada é p, então temos $\mathbf{|y - x| / |y| \leq p}$. Os representantes são escolhidos a partir de uma **varredura nos dados ordenados**, atribuindo-se, de forma gulosa, um representante, para cada grupo de dados relacionados. Na ordenação, um grupo é um conjunto de de números tal que a diferença entre o maior deles, z, para o menor, y, obedece a seguinte relação: $z-y \leq 2p$. Observe que não foi levada em conta a possibilidade de termos valores negativos, quando então seria necessário uma pequena modificação no procedimento.

Para exemplificar, se tivéssemos uma perda aceitável p de **1%** e um conjunto de 9 valores iniciais: **{1.00000 , 1.02345 , 1.06545 , 1.09998 , 1.11678 , 1.11898 , 1.38095 , 1.42349, 1.42095}**, obteríamos três grupos A = **{1.00000, 1.02345, 1.06545, 1.09998}**, B = **{1.11678, 1.11898}** e C = **{1.38095, 1.42349, 1.42095}**, cujos representantees respectivos seriam **1.10526, 1.23433** e **1.52631.**

Terminada a fase de recodificação, temos um dicionário de representantes com suas respectivas frequências, que serão utilizados na fase de compactação.

2.4 Compactação

A segunda fase do método consiste, inicialmente, na **criação da árvore de Huffman [Huffman (1951)],** a partir da estatística gerada para os representantes, na fase anterior. A codificação atribuída a cada representante é, então, anexada ao dicionário guardado.

A geração do arquivo compactado dá-se pela leitura de cada dado original e sua **substituição** pela codificação gerada para seu representante. O dicionário de representantes foi criado usando-se uma árvore rubro-negra **[Szwarcfiter(1994)].**

2.5 Descompactação dos Dados

O processo de descompactação utilizado foi o processo normal do método de Huffman, com a utilização da árvore criada. No presente trabalho não se guardou a árvore propriamente dita, mas o dicionário de representantes, de tal forma que, na etapa de descompactação, inicialmente é recriada a árvore de Huffman, usando esse dicionário.

3. Resultados Computacionais

A seguir apresentamos e discutimos os **resultados computacionais** obtidos com o método. Iniciaremos com a análise dos **tamanhos dos arquivos** após aplicação do processo de compactação, seguida da discussão relativa aos tempos de execução de cada etapa do método.

Não existe uma única medida para a taxa de compactação. Podemos identificar três medidas distintas com esse objetivo, descritas na literatura sobre compactação **[Salomon (2009)],[Sayood (2000)],[Witten (1999)]:** a primeira é a **razão entre entre os tamanhos** do arquivo compactado e o original; a segunda é a **razão entre o espaço economizado e o tamanho do arquivo original** e a terceira é uma média de **bits por caracter** compactado.

No presente trabalho, adotamos o primeiro tipo de medida, ou seja, a **razão entre os tamanhos do arquivo final e do inicial**, expressa em termos percentuais. Quanto menor essa razão, maior a compactação obtida. Por exemplo, uma razão de 10% indica que o arquivo compactado passou a ocupar apenas 10% do volume do arquivo original. Incluímos no tamanho do arquivo final os dados necessários para a compactação, ou seja, o dicionário de representantes.

3.1 Compactação obtida

Para aplicação do método, utilizamos arquivos de tamanhos distintos de forma que pudéssemos fazer uma correlação entre esses tamanhos. Trabalhamos com 5 amostras na faixa de 100 MB a 2 GB **(100MB, 200MB, 500MB, 1GB e 2GB)** e com 9 parâmetros de perda **(0%, 1%, 2%, 3%, 5%, 10%, 15%, 20% e 30%)**. A situação de 0% de perda consiste em compactar o arquivo utilizando apenas o método original de Huffman. Valores de perda acima de 30% não têm sentido para as aplicações reais.

As **Figuras 3.1 e 3.2** apresentam os resultados de compactação obtidos. A escala horizontal indica as perdas, com 9 valores na faixa de 0% a 30% e, na escala vertical o tamanho dos arquivos originais, com 5 valores na faixa de 100MB a 2 GB. Em cada ponto da tabela é mostrada a **taxa de compactação**. A Figura 3.1 apresenta **resultados numéricos** e a Figura 3.2, os mesmos resultados em **forma gráfica**, para melhor visualização.

Perda / Tamanho	0%	1%	2%	3%	5%	10%	15%	20%	30%
100MB	51,4%	18%	16,3%	15,5%	14,5%	13,2%	11,8%	11,1%	9,9%
200MB	51%	18,1%	16,4%	15,5%	14,5%	13,2%	11,8%	11,2%	9,9%
500MB	51,4%	18,1%	16,4%	15,5%	14,5%	13,2%	11,8%	11,1%	9,9%
1GB	50,2%	17,6%	16%	15,2%	14,1%	12,9%	11,5%	10,9%	9,7%
2GB	50%	17,6%	16%	15,1%	14,1%	12,9%	11,5%	10,8%	9,7%

Figura 3.1 – Tabela das Taxas de Compactação

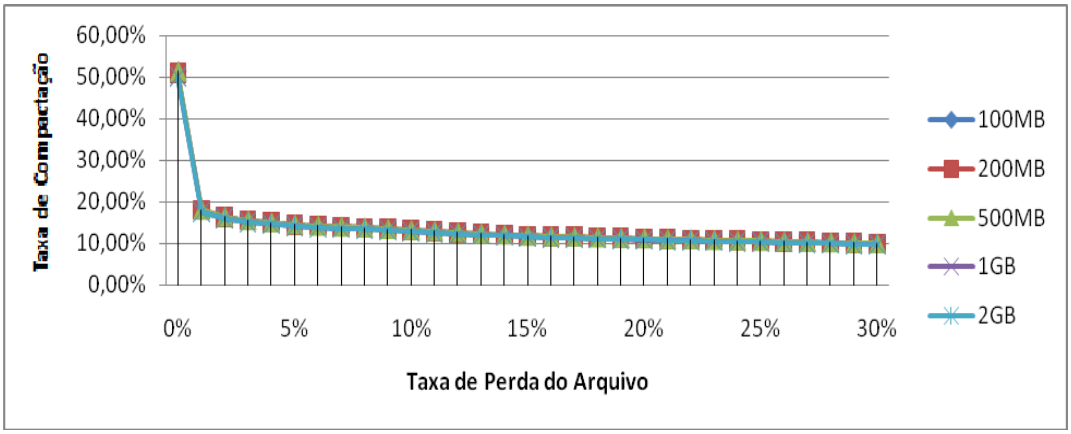

Figura 3.2 – Gráfico das Taxas de Compactação

Uma primeira observação sobre os resultados é que a **taxa de compactação não parece depender do volume de dados original**. Observando as curvas do gráfico da Figura 3.2, vemos que todas elas se superpõem uniformemente. Ou seja, a taxa de compactação parece **depender unicamente da perda adotada,** dada uma distribuição dos dados de entrada.

Temos uma taxa de **compactação decrescente** à medida que a perda aumenta, parecendo tender para um valor próximo a 10%. E temos uma queda brusca na taxa de compactação quando passamos da situação sem perda para aquela de perda de 1%. Essa queda é de cerca de 50% para 20%.

Um resultado **altamente expressivo** é que a compactação obtida é muito **grande** mesmo para a menor perda não nula analisada, de 1%. Neste caso, a taxa é próxima a 20%. Ou seja, o tamanho do arquivo compactado **cai para 1/5** do tamanho inicial.

Vale a pena também analisar o caso **sem perda,** que, embora não seja o objetivo do método desenvolvido, ressalta os expressivos resultados conseguidos. Na situação sem perda, a **compactação foi de 50%**, ou seja, o tamanho do arquivo compactado é a metade do inicial. Esse valor está em **concordância com a literatura** da área de compactação de dados sísmicos **[Xie (2009)]**. Observe-se que parte da compactação obtida corresponde a uma melhor representação dos números envolvidos, quando se trabalha com uma codificação de tamanho variável. A representação em 32 bits dos dados iniciais, feita em ponto flutuante, não está originalmente otimizada. Como só temos 10^8 valores diferentes, eles poderiam ser representados de maneira mais econômica em 27 bits. Outro ponto a notar é que a compactação obtida é muito próxima àquela quando se compacta texto, usando os caracteres como base da compactação, conforme descrito em **[Witten (1999)].**

Desconhecemos qualquer outro método de compactação de dados sísmicos, onde se tenha uma **perda controlada**, como é o caso do método aqui exposto, e que permita uma compactação tão expressiva. Ou seja, o resultado obtido é **bastante promissor** e indica a validade de se buscar aperfeiçoamentos e uma sintonia fina do método desenvolvido.

3.2 Tempo de execução

Nesta seção discutiremos a **performance computacional** do método de compactação proposto, exibindo resultados referentes aos tempos de execução obtidos em cada uma das etapas que o compõem.

Inicialmete, cabe relatar o **ambiente computacional** que deu suporte à aplicação do método. Foi usado um processador AMD Turion(tm) X2 Dual Core Modelo RM-72 com clock de 2.10 GHz, memória de 4 GB e sistema operacional Windows Vista, em uma arquitetura de 64 bits.

Para mensurar o tempo demandado por cada processo, foi utilizada uma **biblioteca padrão** da linguagem C, a time.h. Ela fornece funções como a time() e a difftime() que nos permitem calcular o intervalo de tempo utilizado para executar determinado algoritmo junto a

uma dada entrada de dados. Para diminuir a imprecisão desse tipo de medida, **cada caso foi rodado 9 vezes** e apresenta-se a média de tempo obtida.

As **Figuras 3.3 e 3.4** apresentam os resultados. Na Figura 3.3, as colunas referem-se às **fases** descritas na Seção 2: reorganização dos dados (fase subdividida nos processos de ordenação, por sua vez subdividido em dois: **Seleção por Substituição** e **Intercalação Balanceada**, e **Recodificação dos Dados**), e **Compactação**. Foi colocado também o processo de **Descompactação** que, embora não seja propriamente uma etapa do método, deve ser avaliado. O eixo vertical contém os diversos **tamanhos** de dados de entrada considerados. Os tempos mostrados na tabela correspondem ao pior caso, de perda de 1%, onde o número de representantes é o maior para as perdas estudadas, desconsiderando o caso de 0%. O gráfico da **Figura 3.4,** representa os mesmos dados, tendo como eixo vertical o tempo, em segundos e o eixo horizontal os tamanhos dos arquivos de entrada, apresentando uma **curva** para cada fase do processo.

Tamanho	Seleção por Substituição	Intercalação Balanceada	Recodificação dos Dados	Compactação	Descompactação	Total
100 MB	46s	**0s**	6s	27s	8s	87s
200 MB	96s	**0s**	12s	53s	17s	178s
500 MB	251s	**0s**	29s	135s	44s	459s
1 GB	503s	369s	67s	274s	89s	1302s
2 GB	988s	1099s	137s	559s	178s	2961s

Figura 3.3 – Tabela de Tempo de Execução

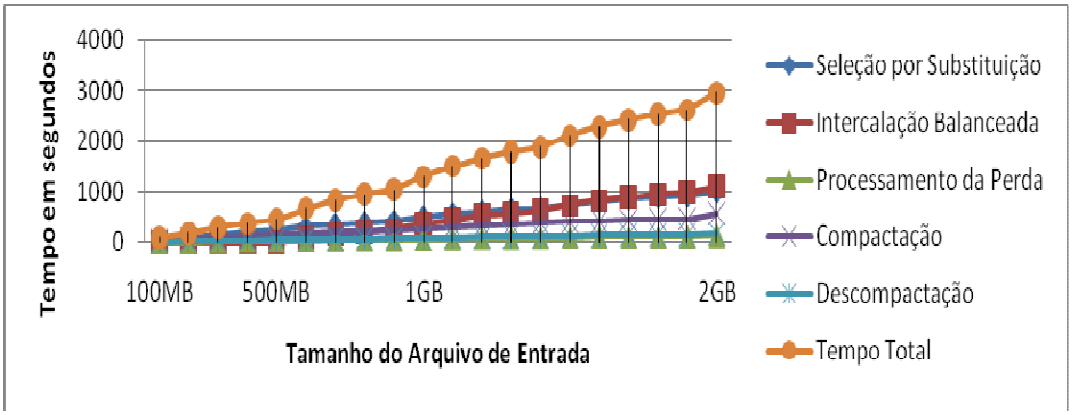

Figura 3.4 – Gráfico de Tempo de Execução por Processo

A razão de termos apresentado apenas os tempos para a perda de 1% é que esses são os **piores casos**. De fato, para um mesmo arquivo a ser compactado, existe uma variação no tempo de execução, considerando-se a porcentagem de erro permitida. Quanto **maior essa porcentagem**, **menor o número de representantes** gerados pelo processo de Recodificação dos Dados, e consequentemente menor o tempo de execução das últimas etapas (Recodificação, Compactação e Descompactação). A fase de **ordenação não é afetada** pela perda.

Pode-se observar que, para arquivos de entrada de até 500 MB, **não foi necessária a ordenação externa**. Neste caso, o tempo total de execução é dominado pela ordenação interna (etapa de Seleção por Substituição), que apresenta um custo computacional em torno de 60% do tempo total. Para arquivos maiores que 500 MB, faz-se necessária a aplicação da **ordenação externa** (Seleção por Substituição + Intercalação Balanceada), sendo que as duas etapas de ordenação apresentaram custos computacionais semelhantes. De qualquer forma, **a ordenação domina todo o processo**, especialmente devido aos altos tempos de E/S.

Em relação à complexidade assintótica do método, os dados são **compatíveis com a complexidade teórica** da ordenação, que é de **O(n log n)** quer a ordenação seja interna ou

externa. Para a etapa de Recodificação dos Dados a complexidade do processo é **O(n),** pois utilizamos uma **abordagem gulosa** para a escolha dos representantes. Na Compactação temos, inicialmente a **criação da árvore de Huffman**, cuja complexidade é **O(k log k)**, onde k é o número de representantes obtidos. Depois temos a **substituição de cada valor inicial pela codificação de seu representante**, feita em **O(log k),** para cada valor já que é usada a árvore rubronegra para isso. Portanto a complexidade é **O(n log k)** e essa complexidade **domina o processo.** Na descompactação temos uma complexidade análoga.

Como as etapas de **ordenação demandam a maior fatia de tempo** de todo o método, é importante que haja esforços no intuito de amenizá-las. Podemos perceber uma tendência no aumento do tamanho da memória principal nos computadores atuais. Inclusive, já existem computadores para fins comerciais com memória RAM de 1 TB. Esse aumento propicia o uso de buffers cada vez maiores e menos gastos com operações de E/S, o que nos leva a considerar que essa fatia de tempo irá diminuir com o passar dos anos e com o desenvolvimento tecnológico.

4. Conclusões

O objetivo principal deste trabalho consistiu do **desenvolvimento e análise** de um método para compactação de dados oriundos de leventamentos sísmicos. O método de compactação proposto é composto por duas etapas: uma primeira de **recodificação com perda dos dados de entrada**, com o intuito de aumentar a redundância dos dados originais, e uma segunda etapa, de **codificação sem perda**, objetivando compactar os dados gerados pela primeira etapa, substituídos por representantes adequados.

Em particular, a primeira etapa do método consiste na construção de uma **função de mapeamento,** que permite associar cada símbolo pertencente ao alfabeto original a um símbolo de um novo alfabeto construído. Cada símbolo deste novo alfabeto é denominado um ***representante*** dos símbolos do alfabeto original. Por sua vez, todo novo alfabeto é dito um ***conjunto de representantes*****.** A função de mapeamento de um símbolo original em seu representante permite erros de representação, que não devem ultrapassar uma porcentagem parametrizada pelo usuário. Mais precisamente, **dado um erro percentual máximo *p*,** definido pelo usuário do método, a função de mapeamento deve garantir que todo valor original v seja mapeado para um representante cujo valor esteja compreendido entre ***(1-p)v e (1+p)v***. Cabe ressaltar que, uma vez que, no problema de compactação de dados sísmicos, tanto o alfabeto original quanto o conjunto de representantes são **valores reais** (representados computacionalmente por tipos de ponto flutuante), as relações de ordem "menor que" e "maior que" estão bem definidas.

A estratégia adotada no trabalho para resolver o problema de determinar um conjunto de representantes e uma função de mapeamento correspondente que aumente a redundância dos dados foi a de **transformar este mesmo problema** em outro mais bem posto, no qual uma função objetivo é especificada. Em particular, transformamos este problema no de **selecionar um conjunto de representantes com cardinalidade mínima**, que permita estabelecer uma função de mapeamento (a qual deve ser explicitada) que respeite a restrição de que todo elemento do alfabeto original seja mapeado em um representante de forma a não ultrapassar erro percentual máximo *p*, pertencente à instância do problema. A solução empregada aqui para resolver este último problema consistiu em **ordenar o conjunto original de símbolos** (valores reais) e realizar uma **busca linear para seleção dos representantes,** seguindo uma estratégia similar a de um algoritmo guloso.

No que se refere à segunda etapa de compactação sem perda, empregou-se o bem **estabelecido método de Huffman [Huffman (1951)].** Este método recebe como entrada uma distribuição das probabilidades de símbolos de um dado alfabeto e fornece como saída uma recodificação livre de prefixos destes símbolos. Tal recodificação **minimiza o valor esperado do tamanho de um arquivo** que segue a distribuição de probabilidades fornecida. O método de

Huffman é ótimo dentre todas as possíveis recodificações livres de prefixo, podendo ser implementado em complexidade de tempo **O(*n* log *n*)** e com complexidade de espaço **O(*n*),** sendo *n* a quantidade de símbolos pertencentes ao alfabeto de entrada.

A implementação do método de Huffman adotada aqui segue a filosofia das estratégias clássicas para implementação do método, que se baseiam na utilização de uma **fila de prioridades** como estrutura de dados. Um ***heap*** é utilizado neste sentido, permitindo inserir um elemento ou remover o elemento de maior prioridade de uma árvore binária representada computacionalmente por um array simples, em tempo O(log *n*). Cabe aqui ressaltar que, caso os símbolos do alfabeto fossem fornecidos ordenados de acordo com suas distribuições de probabilidade, a árvore de Huffman poderia ser construída em tempo O(*n*) utilizando duas filas.

Inicialmente, pudemos observar o interessante resultado de que **a taxa de compactação parece não depender do volume de dados**. De fato, conforme pode ser observado pelos resultados expressos na Figura 3.2, da Seção 3.1, **a taxa de compactação parece depender unicamente da perda adotada**, dada uma distribuição dos dados de entrada (para este trabalho, em particular, descrita na Seção 2.1).

Observa-se ainda um **decréscimo evidente na taxa de compactação** à medida que o erro permitido aumenta. Em particular, temos como resultado uma expressiva queda na taxa de compactação quando passamos de um cenário de compactação sem perda para um com perda mínima, de 1%. Neste caso, a queda é de cerca de **50% para 20%**.

Desconhecemos qualquer outro método de compactação de dados sísmicos, onde se tenha uma **perda controlada**, como é o caso do método aqui exposto, e que permita uma **compactação tão expressiva**. Tais resultados computacionais obtidos permitem-nos classificar o novo método proposto como consideravelmente promissor, e validam a busca por futuros aperfeiçoamentos, incentivando uma sintonia fina do trabalho executado.

Em relação ao **tempo de execução** decorrente de aplicações do método, analisando-se os resultados apresentados nas Figuras 3.3 e 3.4, presentes na Seção 3.2, pode-se observar que o tempo total de execução é **dominado pela etapa de Seleção por Substituição** para arquivos de até 500 MB, o qual apresenta um custo computacional em torno de **60%** do tempo total. Para arquivos maiores que 500 MB, faz-se necessária a aplicação do método de **Intercalação Balanceada,** o qual apresenta custo computacional compatível ao da fase se Seleção por Substituição. Extrapolando estes resultados para arquivos maiores, os quais possivelmente não caberão em memória, observa-se uma tendência clara de concentração do **gargalo do tempo de execução nas fases de Intercalação Balanceada e Seleção por Substituição**, as quais compõem o processo de ordenação do dado de entrada e possuem custo computacional similar.

No que se refere à **complexidade assintótica de pior caso do método**, observa-se para a etapa de Ordenação dos dados de entrada, uma complexidade de tempo limitada por **O(n *log* n),** onde *n* é a quantidade de amostras no arquivo a ser ordenado. Na etapa de Recodificação dos Dados, no entanto, temos uma **complexidade inerentemente linear**, pois utilizamos uma **abordagem gulosa** para a escolha dos representantes, que percorre o vetor ordenado uma única vez, selecionando os representantes correspondentes. Nas etapas de **Compactação e Descompactação**, a complexidade é basicamente a da aplicação do método de Huffman para a massa de dados, o que resulta em uma complexidade **O(n *log* k)** tanto para a Compactação quanto para a Descompactação, onde *n* é a quantidade de amostras no arquivo a ser compactado e *k* é a quantidade de representantes presentes na árvore de Huffman. Pelas complexidades assintóticas descritas anteriormente, e devido ao fato das constantes envolvidas na complexidade de tempo de cada etapa do algoritmo serem relativamente baixas e similares, justifica-se o fato da etapa de Ordenação dominar o tempo de execução experimental do método.

Em relação aos **trabalhos futuros** decorrentes, uma primeira questão em aberto seria a da **determinação de um algoritmo ótimo para o problema da seleção de representantes**. O algoritmo guloso aqui proposto mostrou-se experimentalmente adequado, atingindo bons resultados na prática. No entanto, seria interessante **demonstrar formalmente** que o mesmo determina sempre soluções ótimas para o problema, ou ainda projetar um novo algoritmo que resolva o problema em sua otimalidade. Uma segunda questão a ser naturalmente explorada

seria a **realização de experimentos computacionais com dados sísmicos reais**. Por questões de confidencialidade, este projeto lidou apenas com dados criados sinteticamente a partir de uma distribuição de probabilidades, que possui comportamento similar a de um dado sísmico. A realização de experimentos com dados reais nos permitiria analisar o impacto de variações na distribuição de probabilidades na taxa de compactação obtida pelo método. Por fim, uma **terceira questão em aberto,** que seria mais abrangente e impactante, seria **demonstrar que o método proposto é ótimo** dentre todas as possíveis recodificações livres de prefixo nas quais cada símbolo de um alfabeto original pode ser mapeado em um símbolo de um novo alfabeto, desde que tal mapeamento **respeite um erro percentual máximo *p***, pertencente à instância do problema. Este seria um forte resultado teórico, o qual resultaria em uma **extensão da otimalidade** atingida pelo algoritmo de Huffman, dentre as compactações sem perda livres de prefixo, para compactações livres de prefixo com perda controlada *p*.

Referências

Anjos, F. M. *Reorganização e compressão de dados sísmicos***,** Dissertação (Mestrado em Informática) - Pontifícia Universidade Católica do Rio de Janeiro, 2007.

Averbuch, A.Z.; Meyer, F.; Stromberg, J.O.; Coifman, R;Vassiliou, A.; Low Bit-Rate Efficient Compression for Seismic Data, in *IEEE Transactions On Image Processing*, Vol. 10, n. 12, pp 1801-1814, 2001**.**

Cormen, T. H.; Leiserson, C.; Rivest R. L.; Stein, C. *Introduction to Algorithms*, 3rd Ed., MIT. 2009.

Huffman, D.A. ***A Method of Construction of Minimum Redundancy Codes,*** in *Procedings of the IRE* - 10 pp. 1098-1101, 1951**.**

Panetta, J. et al; Accelerating Kirchhoff Migration by CPU and GPU Cooperation**.** in *SBAC-PAD*, pp. 26-32, 2009.

Salomon, D. *Data Compression The Complete reference*, Springer Verlag. 4th Ed. 2007.

Sayood K. *Introduction to Data Compression*, 2nd Ed., Morgan Kaufmann Pub. 2000**.**

Sheriff, R. E.; Geldart, L. P. *Exploration Seismology.* Cambridge University Press, United Kingdom, 1995.

Szwarcfiter, J; Markenzon, L. *Estruturas de Dados e seus Algoritmos*. LTC Editora, 2a. Ed., 1994.

Villasenorl, J.D.; Ergas, R.A; Donoho, P.L. Seismic Data Compression Using High-Dimensional Wavelet Transforms**,** in *Proc. Data Compression Conf.*, IEEE Computer Society Press, pp. 396-405, 1996.

Vitter, J. S. *Algorithms and Data Structure for External Memory,* Series on Foundations and Trends in Theoretical Computer Science, now Publishers, Hanover, MA, 2008.

Witten, I.H; Moffat, A.; Bell, T. *Managing Gigabytes – Compressing and Indexing Documents and Images*, Academic Press. 2nd Ed. 1999.

Xie, X.; Qin, Q. Fast Lossless Compression of Seismic Floating-Point Data, in *Proc of International Forum on Information Technology and Applications*, pp. 235-238, 2009**.**