COMPRESSÃO DE DADOS SÍSMICOS ATRAVÉS DA REESTRUTURAÇÃO DE ARQUIVOS SEG-Y

Gustavo Rodrigues Lima

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Jano Moreira de Souza
Geraldo Zimbrão da Silva

Rio de Janeiro
Março de 2015

COMPRESSÃO DE DADOS SÍSMICOS ATRAVÉS DA REESTRUTURAÇÃO DE ARQUIVOS SEG-Y

Gustavo Rodrigues Lima

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Geraldo Zimbrão da Silva, D.Sc.

Prof. Alexandre de Assis Bento Lima, D.Sc.

Prof. Silvana Rossetto, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2015

Lima, Gustavo Rodrigues

Compressão de dados sísmicos através da reestruturação de arquivos SEG-Y/Gustavo Rodrigues Lima. – Rio de Janeiro: UFRJ/COPPE, 2015.

XI, 85 p.: il.; 29, 7cm.

Orientadores: Jano Moreira de Souza

Geraldo Zimbrão da Silva

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2015.

Referências Bibliográficas: p. 79 – 85.

1. Compactação de Dados Sísmicos. 2. Compactação sem Perdas. 3. SEG-Y. 4. Reestruturação de SEG-Y. I. Souza, Jano Moreira de *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Agradecimentos

Acredito que a vida é feita de escolhas e nós somos o resultado das escolhas que fazemos. Algumas muito difíceis e outras mais fáceis. Algumas que levam por caminhos sinuosos e estreitos, e outras que levam a atalhos. Mas, independente de qual caminho seguir, é muito mais fácil quando temos o apoio daqueles que amamos. Por isso, gostaria de dedicar essa conquista a algumas pessoas que foram fundamentais ao logo deste ciclo.

Gostaria de agradecer aos meus orientadores, prof. Jano Moreira de Souza e prof. Geraldo Zimbrão, que dedicaram seu tempo a orientar este trabalho, sempre com sugestões e ideias. Também quero agradecer aos membros da banca que dedicaram seu tempo a analisar esta dissertação.

Agradeço à ANP (Agência Nacional do Petróleo, Gás Natural e Biocombustíveis) e ao BDEP (Banco de Dados de Exploração e Produção) que disponibilizaram os dados sísmicos reais utilizados nesta pesquisa.

Gostaria também de agradecer ao apoio da família que sempre esteve presente. Especialmente aos meus pais Sonia e Ciraldo que lutaram para que eu tivesse uma educação de qualidade e me auxiliam principalmente nas dificuldades.

Quero agradecer aos meus amigos que foram capazes de entender e desculpar minha ausência, pelas conversas, pelos incontáveis cafés, pelas cervejas do final de semana, entre muitos outros momentos. Amigos que sempre se mostraram preocupados e dispostos a ajudar quando mais precisei. Sempre contribuindo com boas ideias e com momentos de descontração e de boas risadas – extremamente importante, diga-se de passagem.

Agradeço aos professores do PESC que contribuíram com minha formação acadêmica, ministrando aulas de qualidade. Também agradeço aos funcionários do PESC e da CAPES pela infraestrutura que tive acesso. Foram eles que, desde o cafezinho até os mais complexos processos da secretaria, promoveram um ambiente propício ao desenvolvimento pessoal e acadêmico.

Finalmente, agradeço a Deus pela força que tem me dado para superar os obstáculos e as dificuldades, e a Nossa Senhora pela sua divina intercessão.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

COMPRESSÃO DE DADOS SÍSMICOS ATRAVÉS DA REESTRUTURAÇÃO DE ARQUIVOS SEG-Y

Gustavo Rodrigues Lima

Março/2015

Orientadores: Jano Moreira de Souza
Geraldo Zimbrão da Silva

Programa: Engenharia de Sistemas e Computação

Através da aquisição, processamento e interpretação de dados sísmicos, é possível identificar as diferentes camadas que compõem a superfície terrestre, bem como identificar suas propriedades e características. Estas técnicas também podem ser utilizadas na localização de novas reservas de combustíveis fósseis, para monitorar os níveis das reservas conhecidas e para determinar o melhor local de perfuração de um poço de petróleo que permita uma extração mais eficiente. Contudo, o tamanho dos dados sísmicos aumentou significativamente, principalmente com o surgimento de novas técnicas de aquisição 3D e 4D, dificultando os processo de transmissão e armazenamento.

Para superar esta questão, as organizações podem recorrer às inúmeras técnicas de compressão de dados presentes na literatura ou aos compactadores existentes no mercado. Um dos compactadores mais eficientes em tempo de compressão, o PBZip2 oferece desempenho variável quanto às taxas de compressão, dependendo do estágio de processamento do dado sísmico. Este trabalho propõe, então, formas de reestruturar os dados sísmicos de maneira a aumentar as taxas de compressão e reduzir o tempo de processamento do PBZip2. Ademais, propõe-se uma abordagem de compressão sem perdas que englobe a reestruturação do dado durante a compressão e reconstrução do dado original durante a descompressão. Desta forma, é possível reduzir o tamanho do arquivo comprimido em até 20%, sem causar danos ou perdas no dado sísmico.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

SEISMIC DATA COMPRESSION BY REESTRUCTURING SEG-Y FILES

Gustavo Rodrigues Lima

March/2015

Advisors: Jano Moreira de Souza
Geraldo Zimbrão da Silva

Department: Systems Engineering and Computer Science

Through seismic data acquisition, processing and interpretation, it is possible to identify the layers of Earth's surface, as well as identify their properties and characteristics. These techniques may also be used to localize reservoirs of fossil fuels, to monitor the level of known reservoirs, and to determine the best location to drill the oil, allowing an efficient extraction of it. However, the size of seismic data has been increased, mainly with the development of new 3D and 4D seismic data acquisition techniques, which hinder the processes of data transmission and storage.

To overcome this issue, the organizations may appeal to several data compression techniques or to several compactors available nowadays. One of the most time-efficient compactors, the PBZip2 offers variable performance in compression ratio, depending on the stage of processing of seismic data. This work proposes some forms to restructure the seismic data in order to increase the compression ratio and reduce the execution time of PBZip2. Moreover, it proposes a lossless compression scheme that comprises the restructuration of seismic data during compression and rearrangement of the original structure in decompression process. Therefore, it is possible to reduce the size of compressed files in 20%, without causing neither damage nor loss in seismic data.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contexto

No fim da década de 40 e início da década de 50, o petróleo impulsionou um dos maiores movimentos da história do Brasil, denominado "O Petróleo é Nosso", que foi marcado por passeatas, comícios e debates. Como resultado deste processo histórico, em 1953 fundou-se a Petrobras, criada com a Lei do Petróleo e que é um dos maiores nomes mundiais na exploração de hidrocarbonetos. Atualmente, a estatal possui técnicas e tecnologia necessárias à exploração de petróleo no pré-sal.

Não apenas a Petrobras, mas também outras organizações ligadas à exploração de combustíveis fósseis vêm aperfeiçoando e pesquisando novas técnicas que auxiliam no processo de localização e extração destes combustíveis. Como os processos de perfuração do poço e extração do óleo são dispendiosos, é necessário um bom planejamento antes de qualquer ação, ou seja, é necessário avaliar se existe combustível fóssil no subsolo, determinar o melhor local de perfuração para maximizar a extração, entre outros. Para realização destas e de outras ações, as organizações "coletam" e analisam dados sísmicos da região e utilizam diversas técnicas para processá-los e analisá-los.

Os dados sísmicos são dados que refletem as características e propriedades da superfície da Terra e de suas diversas camadas. Eles são obtidos por meio de um processo que envolve a produção de energia que, ao ser refletida pelas diversas camadas da superfície terrestre, é captada por sensores e gravada. Este "dado sísmico bruto" passa por um exaustivo processamento que, entre muitos outros objetivos, visa a remover ruídos, realçar as características sísmicas e remover "falsas regiões". Após o processamento, o dado sísmico está pronto para ser analisado e interpretado.

Desta forma, ele pode ser utilizado para diversos fins, como monitorar o nível dos reservatórios, determinar a existência de hidrocarbonetos no local, ou até mesmo para pesquisas relacionadas às características geográficas da região em questão. De-

pendendo da finalidade do dado, as etapas de processamento e seus parâmetros de configuração podem variar, justamente para evidenciar determinadas características.

Quanto à estrutura do dado sísmico, ressalta-se que cada empresa pode adotar aquela que mais a convier, não havendo necessariamente uma padronização. Contudo, para fins de troca de dados sísmicos entre diferentes organizações, a *Society of Exploration Geophysicists* (SEG)[1] propôs a padronização do dado através do formato conhecido como *SEG-Y*.

Independente de sua estrutura, um grande desafio quando se fala em dado sísmico é reduzir seu tamanho para armazenamento ou transmissão, tendo em vista que ele pode ultrapassar a ordem de terabytes – principalmente com as novas técnicas de aquisição de dados sísmicos 3D e 4D que produzem dados volumosos. Para tal, é possível recorrer às inúmeras técnicas de compressão de dados (genéricos ou sísmicos) presentes na literatura ou até mesmo aos compressores existentes no mercado.

1.2 Motivação

Os avanços tecnológicos são inegáveis, sobretudo no que diz respeito à capacidade de armazenamento e na velocidade de transmissão de dados via Internet. Contudo, lidar com dados cujo tamanho pode superar a ordem de terabytes é um processo desafiador, mesmo com o apoio de *storages* e redes cada vez mais rápidas. Por essa razão, o emprego de técnicas de compressão de dados é essencial e traz diversos benefícios no que tange ao gerenciamento dos dados sísmicos. Entre os benefícios citam-se a diminuição no tempo de transmissão e o aumento na capacidade de armazenamento dos *storages*, visto que o tamanho do dado comprimido é inferior ao dado sem compressão.

A literatura é vasta no que diz respeito às técnicas de compressão de dados sísmicos, sobretudo quando a perda é admissível. Por outro lado, a aplicação de técnicas de compressão sem perdas é mais aceita pela comunidade de especialistas em geofísica, apesar de possuírem desempenho inferior às técnicas de compressão com perdas. Ainda assim, a quantidade de estudos relacionados à compressão sem perdas é bem inferior do que aqueles centrados em compressão com perdas – ou seja, ainda há espaço para mais pesquisas que envolvam compressão sem perdas.

Além disso, a grande quantidade de compressores existentes do mercado pode vir a se tornar uma solução quando se busca simplicidade, praticidade e baixo custo. Geralmente estes programas são utilizados no dia a dia para comprimir arquivos diversos e não apresentam um bom desempenho na compressão de dados sísmicos. Entretanto, é possível melhorar as taxas e os tempos de compressão destes programas através de abordagens que diminuam a entropia do dado sísmico.

[1] `http://www.seg.org`

Portanto, as principais motivações deste trabalho são:

1. As técnicas de compressão são essenciais e somam benefícios no gerenciamento de dados;
2. Existe espaço para mais pesquisa, principalmente relacionado à compressão de dados sísmicos sem perdas; e
3. É possível utilizar compressores populares na compressão de dados sísmicos de maneira mais eficaz.

1.3 Proposta

Esta dissertação propõe uma alternativa às técnicas de compressão de dados sísmicos sem perdas apresentadas na literatura. Busca-se uma maneira simples de comprimir os dados, porém eficiente tanto em tempo quanto em taxa de compressão. Por essa razão, a abordagem desta dissertação se inspira em um dos compressores mais eficientes em tempo de processamento existentes atualmente: PBZip2[2].

O PBZip2 é a versão paralela do compressor BZip2[3] que utiliza a biblioteca *pthread* e cujo *speed-up* é quase linear. Apesar destes programas serem capazes de comprimir um arquivo por vez, eles apresentam boas taxas de compressão para diversos tipos de dados.

Com relação a dados sísmicos, estes *softwares* de compressão de dados "genéricos" não apresentam um bom desempenho quanto às taxas de compressão. Todavia, é possível melhorá-las através da reestruturação do dado sísmico que, no caso deste trabalho, é representado por arquivos SEG-Y.

Um arquivo SEG-Y é composto por dois tipos de dados, a saber: cabeçalhos e ou amostras. Os cabeçalhos constituem uma pequena parte do dado sísmico e contêm informações relevantes sobre suas características e sobre o processo de aquisição. Já as amostras são números inteiros ou em ponto flutuante que foram digitalizados por sensores durante o processo de aquisição. Elas estão organizadas por traços que representam curvas das camadas de uma região da terra.

Propõe-se, então, a estratégia de tratar cabeçalhos e amostras separadamente durante a compressão. Como o primeiro representa uma pequena parte do arquivo e possui baixa entropia, pode ser comprimido sem nenhum pré-processamento adicional. Já para o segundo, propõe-se separar as amostras de 32 bits em diversas partes, cujo número de bits pode variar, e que devem ser comprimidas separadamente. Um exemplo de reestruturação é separar amostras em expoente (que inclui o bit de sinal), mantissa alta e baixa. Desta forma, os cabeçalhos e as partes das amostras são

[2] `http://compression.ca/pbzip2`
[3] `http://www.bzip.org`

comprimidas separadamente pelos mesmos algoritmos de compressão utilizados pelo PBZip2. Com esta estratégia, é possível diminuir o tamanho do arquivo comprimido em até 20%, em relação ao arquivo comprimido gerado pelo PBZip2.

Para mostrar que a reestruturação é realmente eficaz, esta dissertação apresenta a implementação do SEG-Y-BZ2 que nada mais é do que um compressor de arquivos SEG-Y que utiliza a biblioteca *pthread* – como o PBZip2 –, realiza a reestruturação do arquivo durante o processo de compressão e comprime-o com auxílio da biblioteca *libbzip2*. Por sua vez, o processo de descompressão é encarregado de descomprimir o arquivo comprimido e restabelecer a estrutura original do SEG-Y.

O compressor SEG-Y-BZ2 é uma prova de conceito de que a reestruturação do dado sísmico é de fato eficiente, melhorando a taxa de compressão. Além disso, o processo de reestruturação é invisível ao usuário e não danifica a estrutura do arquivo SEG-Y, visto que a mesma é restaurada durante o processo de descompressão.

1.4 Contribuições

As contribuições desta dissertação podem ser resumidas nos seguintes itens:

1. Uma taxonomia que busca organizar a literatura relacionada à compressão de dados sísmicos, visto que os artigos são numerosos nesta área.

2. A proposta de estratégias de reestruturação do dado sísmico representados em arquivo SEG-Y que efetivamente reduzem a entropia do conjunto de dados.

3. A implementação de um compressor *multithread* de dados sísmicos, cujo formato é SEG-Y, que utiliza a biblioteca *libbzip2* e reduz o tamanho do arquivo comprimido em até 20% em relação ao PBZip2.

4. A prova de conceito de que a reestruturação do dado sísmico, antes ou durante a compressão, pode ser benéfica, reduzindo a entropia do dado sísmico.

5. A avaliação do compressor mediante a realização de experimentos com dados sísmicos reais sem processamento e em diversos estágios de processamento.

1.5 Organização

Esta dissertação está distribuída em cinco capítulos dos quais este é o primeiro. O capítulo 2 apresenta um panorama acerca do dado sísmico, introduzindo os processos de aquisição, processamento e interpretação do dado. Além disso, este capítulo apresenta a taxonomia desenvolvida para organizar a área de compressão de dados sísmicos e descreve os trabalhos relacionados.

No capítulo 3, a proposta de trabalho é elucidada, destacando a fundamentação teórica e as hipóteses investigadas. Também é apresentado o desempenho de diversos programas na compressão de dados sísmicos com a estrutura original do SEG-Y e após reestruturação. Além disso, a arquitetura, funcionamento e interface do compressor de dados sísmicos SEG-Y-BZ2, que implementa a reestruturação e compressão, são descritos.

Já o capítulo 4 traz a descrição dos experimentos realizados e do ambiente de teste. No que diz respeito aos resultados alcançados, estes são divididos em três partes. A primeira compara o desempenho das estratégias de reestruturação propostas e identifica a melhor delas. Já a segundo compara o desempenho da melhor estratégia de reestruturação proposta com algumas estratégias de reestruturação descritas na literatura, enquanto a terceira compara o desempenho do SEG-Y-BZ2 com o desempenho PBZip2 na compressão de dados sísmicos.

Por fim, o capítulo 5 relata as conclusões desta pesquisa e aponta possíveis caminhos a serem explorados como trabalhos futuros.

Capítulo 2

O Dado Sísmico e a Compressão

Este capítulo descreve o dado sísmico que é alvo deste estudo, enfatizando sua aquisição, estrutura e finalidade, bem como os processos que o utilizam. Apresenta-se também o problema alvo deste estudo – ou seja, o tamanho do dado sísmico – e os trabalhos relacionados à compressão de dados sísmicos, organizados pela taxonomia proposta.

2.1 Dado Sísmico

Antes de responder perguntas como "*Como é adquirido?*" e "*Como é o processamento?*", é preciso entender o que é o dado sísmico e o que ele representa. O dado sísmico é uma imagem das camadas da Terra abaixo de sua superfície. Numa analogia grosseira à medicina, é como se a superfície da Terra fosse a pele e as camadas representassem os ossos. Assim, uma imagem em raio-x é o que se poderia chamar de dado sísmico.

Os dados sísmicos podem ser 2D, 3D ou 4D, dependendo da configuração do processo de aquisição. Há algum tempo atrás, a maneira mais comum de obter o dado sísmico era a partir de sensores dispostos em linha reta, gerando a imagem de uma sessão da Terra – este é dado sísmico 2D, ilustrado na Figura 2.1.

Com o passar dos anos, percebeu-se que utilizar linhas paralelas de sensores e igualmente espaçadas era mais vantajoso do que realizar interpolações no dado sísmico 2D. Surgiu, assim, o dado sísmico 3D que representa um volume da Terra e é amplamente utilizado na indústria do petróleo [1].

Finalmente, o dado sísmico 4D mostra o volume da Terra em diferentes instantes de tempo, ou seja, ele é o dado sísmico 3D adicionando-se a dimensão tempo e pode ser empregado na avaliação do tempo de vida de reservatórios [2].

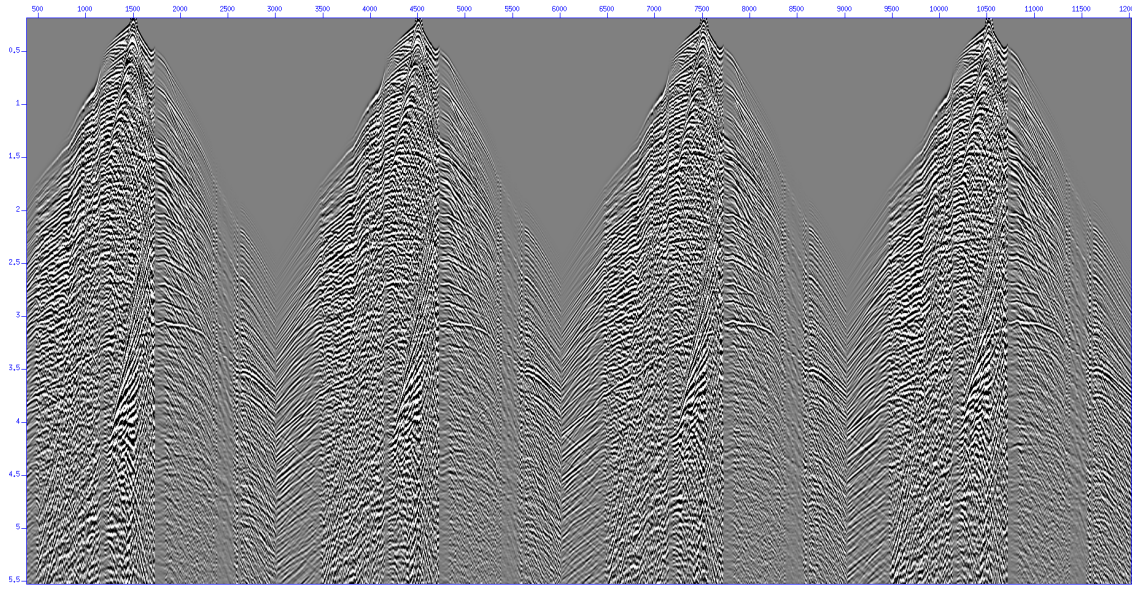


Figura 2.1: Visualização de um dado sísmico 2D gerado pelo programa Seismic Unix.

A escolha de qual processo de aquisição utilizar depende do quão valioso é o dado sísmico face aos custos da aquisição [3]. Há algumas décadas atrás, a aquisição 3D ainda em desenvolvimento era pouco utilizada devido ao seu alto custo, comparando-se com a aquisição 2D. Contudo, a aquisição em três dimensões se tornou um processo praticamente indispensável na indústria do petróleo nos dias de hoje.

Essencialmente, o dado sísmico pode ser armazenado na forma de arquivo que é composto por dois tipos de dados: cabeçalhos e amostras. Os cabeçalhos, que são binários ou textuais, apresentam informações sobre o dado e o processo de aquisição. Já as amostras são números inteiros ou em ponto flutuante que foram digitalizadas por sensores durante o processo de aquisição. Elas estão organizadas por traços que, por sua vez, representam curvas que definem as camadas de uma região da terra. Em um dado sísmico, todos os traços tem a mesma quantidade de amostras.

No que diz respeito às amostras em ponto flutuante, seu formato pode ser IBM ou IEEE 754. A principal diferença entre eles é que o formato IBM apresenta 7 bits de expoente e 24 bits de mantissa, enquanto que o formato IEEE 754 apresenta 8 bits de expoente e 23 bits de mantissa. Somando um bit de sinal em ambos os formatos, tem-se amostras de 32 bits, como pode ser visto na Figura 2.2.

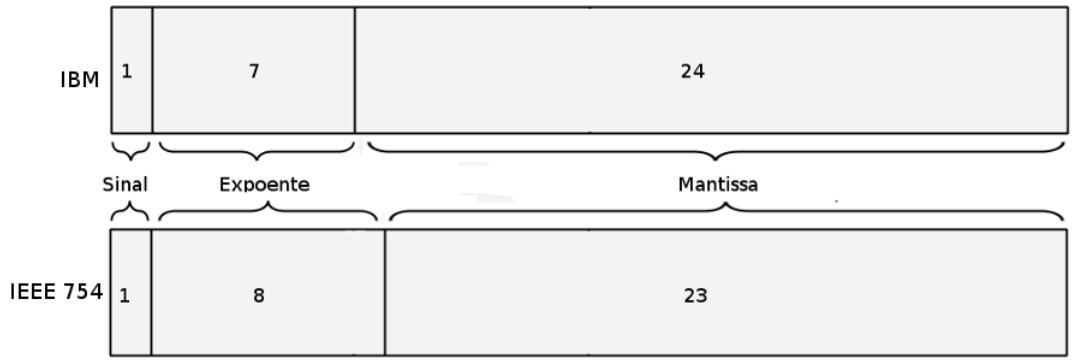


Figura 2.2: Formatos IBM e IEEE 754 de números em ponto flutuante de 32 bits.

quanto à estrutura de arquivo do dado sísmico, cada organização tem autonomia para armazenar o dado no formato que a satisfizer, o que dificulta a troca de dados entre organizações. Foi pensando nisto que a SEG propôs uma estrutura de dados sísmicos conhecida como SEG-Y, amplamente utilizada nos dias de hoje na troca de dados sísmicos.

Como pode ser visto na Figura 2.3, a estrutura do SEG-Y revisão 1 é formada por seis componentes: *optional SEG-Y tape label*, *textual file header*, *binary file header*, *N extended textual file header*, *M trace headers* e *M data traces*.

Dentre estes componentes, destaca-se que o *textual file header* contem informações sobre a empresa responsável pela aquisição do dado sísmico, enquanto que o *binary file header* apresenta informações utilizadas no processamento do dado. Por outro lado, as amostras do dado sísmico estão localizadas nos *data traces*, enquanto que o *trace header* traz informações sobre o traço sísmico.

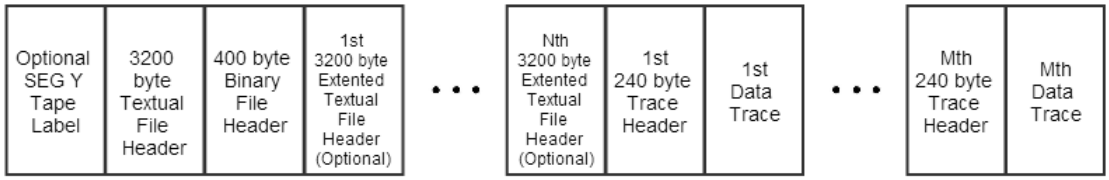


Figura 2.3: Estrutura de arquivos SEG-Y.

Esta dissertação utiliza apenas dados sísmicos no formato SEG-Y cujas amostras são números em ponto flutuante de 32 bits estruturadas no formato IBM. Todavia, as soluções propostas aqui podem ser adaptadas para outros formados de dado sísmico e amostras.

Mas para quê servem os dados sísmicos? Segundo Bacon et al. em [3], o dado sísmico é importante tanto na busca por hidrocarbonetos quanto para se obter uma exploração eficiente destes recursos. É através do dado sísmico que é possível entender a estrutura geológica de uma região, distinguir diferentes camadas, identificar

formações rochosas que armazenam combustíveis fósseis, bem como identificar estruturas que são barreiras à extração do recurso.

Para se chegar a descoberta de óleo ou gás no subsolo, as empresas realizam três processos básicos: aquisição, processamento e interpretação do dado sísmico. Na aquisição, são realizados o planejamento e aquisição do dado sísmico de uma área da superfície terrestre. Durante o processamento, o dado sísmico obtido na aquisição passa por diversos procedimentos cujo objetivo é aumentar sua qualidade e obter uma imagem clara e realista das estruturas geológicas da região. Por fim, o dado é analisado exaustivamente por um geofísico durante a interpretação, que aponta possibilidades da existência de reservatórios de combustíveis fósseis na área.

2.1.1 Aquisição de Dados Sísmicos

A aquisição do dado sísmico pode ser realizada tanto na terra quanto no mar – ou em ambientes híbridos, onde há a transição entre os meios. Essencialmente, o processo é o mesmo nos dois ambientes com diferenças na tecnologia e equipamentos empregados.

Para se estudar a formação da estrutura geológica da Terra, os geofísicos e geólogos analisam ondas sísmicas que podem ser geradas por terremotos ou pela intervenção humana. No processo de aquisição de dados sísmicos para exploração de hidrocarbonetos, as ondas sísmicas são artificialmente geradas por fontes de energia acústica na superfície da Terra e propagam-se para as camadas mais internas da crosta terrestre. À medida que a onda passa por estruturas e camadas com impedância acústica diferentes, parte dela é refletida e capturada por sensores. Os sensores, por sua vez, geram pulsos elétricos que são registrados por sismógrafos e digitalizados [4].

O tempo de propagação da onda também é importante neste processo. Sabendo o tempo de viagem – o tempo desde que a onda foi gerada até o momento em que a onda refletida é capturada pelo sensor –, é possível determinar a profundidade em que ocorreu a reflexão. Consequentemente, tem-se uma imagem da superfície terrestre [4].

Tanto na terra quanto no mar, é importante planejar a aquisição com precaução. É preciso se ater às barreiras naturais que dificultam a disposição de sensores e fontes de energia, minimizar os impactos ao ambiente e seus habitantes, utilizar fontes de energia adequadas e mais eficientes, gravar as posições dos elementos envolvidos na aquisição – sensores, fontes de energia, etc – com precisão, entre outros [3].

No que diz respeito à tecnologia empregada, o processo de aquisição realizado em terra pode utilizar várias fontes de energia geradora da onda, como explosivos, caminhões que derrubam pesos e vibradores sísmicos. Já os sensores que captam as

ondas refletidas são chamados de geofones.

A Figura 2.4 ilustra o processo de aquisição sísmica em terra. Neste caso, uma fonte de energia gera ondas sísmicas que se propagam pelas diversas camadas da superfície terrestre. A medida que a onda atinge a superfície de transição entre as camadas 1 e 2, parte da onda é refletida e parte é refratada. A parte refletida da onda é capturada por geofones localizados na superfície que, por sua vez, transmite os impulsos elétricos para uma unidade responsável por gravá-los – estes impulsos elétricos viram as amostras sísmicas. Já a parte da onda que foi refratada se propaga pela camada 2 até atingir a superfície de separação entre as camadas 2 e 3, que novamente reflete e refrata parte da energia.

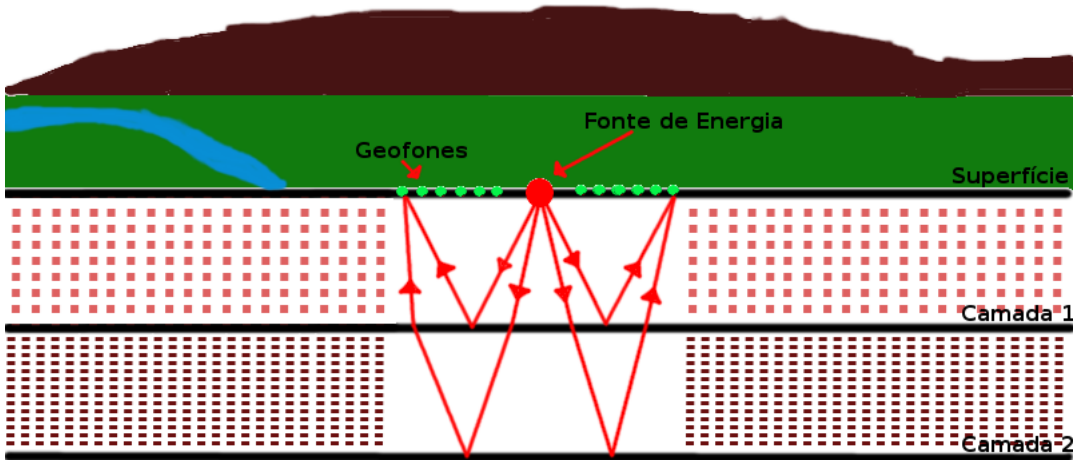


Figura 2.4: Processo de aquisição de dados sísmicos na terra, adaptado da figura proposta pela Universidade Fernando Pessoa[1].

Quanto ao processo de aquisição realizado no mar, é comum utilizar um navio com uma fonte de energia e sensores presos a ele. A fonte pode ser uma arma de ar comprimido, enquanto que os sensores são chamados de hidrofones. A Figura 2.5 mostra o processo de aquisição em alto mar que é muito semelhante ao processo de aquisição em terra. Assim, as ondas sísmicas emitidas pela fonte se propagam pelas diferentes camadas e, a medida que atingem a superfície de transição entre camadas com impedâncias diferentes, parte da onda é refletida e capturada pelos hidrofones. Por sua vez, os dados capturados pelos hidrofones são transmitidos a uma estação e finalmente gravados.

[1] `http://homepage.ufp.pt/biblioteca/Seismic/Pages/Page5.htm`, 05/11/2014

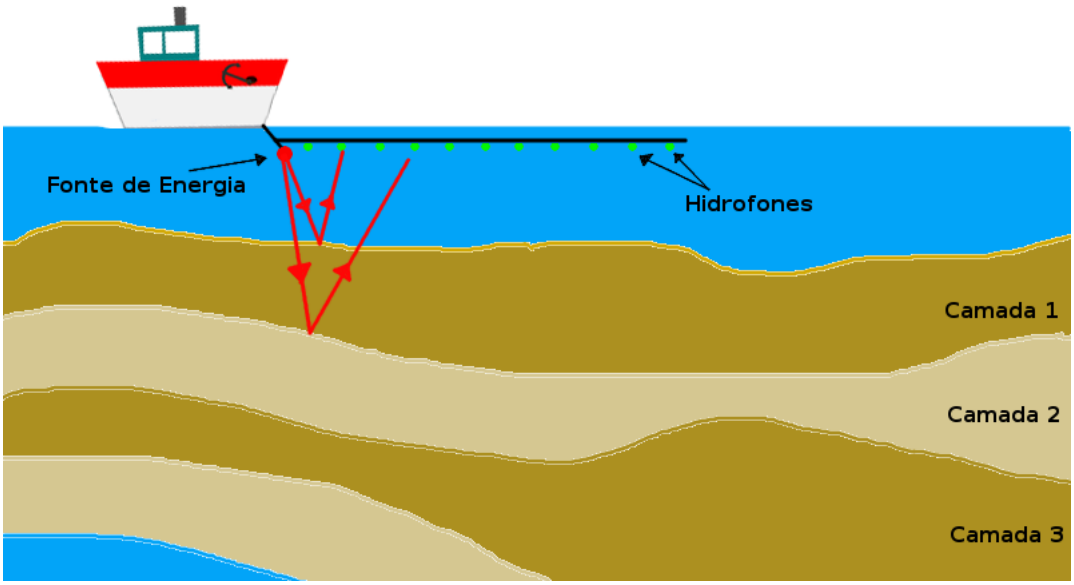
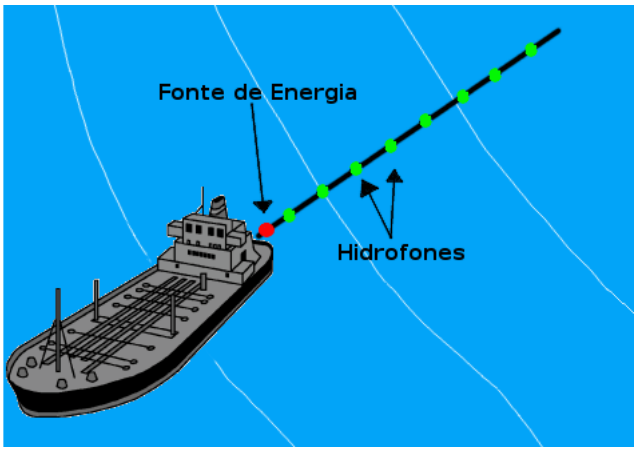


Figura 2.5: Processo de aquisição de dados sísmicos no mar, adaptado da figura proposta por OpenLearn[2].

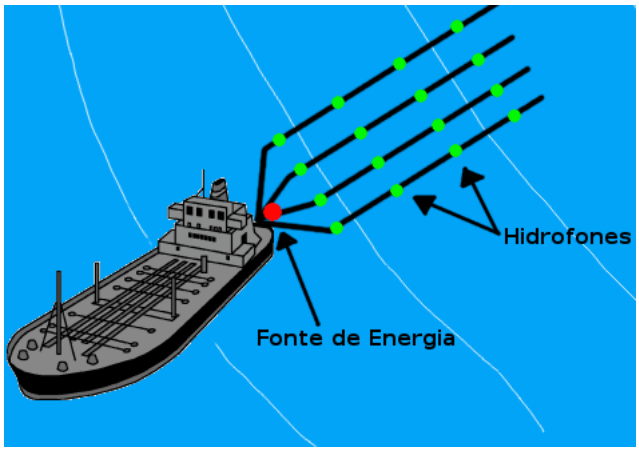
No que diz respeito às dimensões do dado, a aquisição de dados sísmicos 2D utiliza uma linha com vários sensores separados por um *offset*. Esta configuração de aquisição permite capturar uma sessão da Terra. Por sua vez, a aquisição de dados sísmicos 3D utiliza uma malha de sensores – varias linhas paralelas e igualmente espaçadas, com sensores separados por um *offset* [5]. Neste caso, é possível capturar a imagem de um volume da Terra.

A Figura 2.6 ilustra os processos de aquisição sísmica 2D e 3D, este último mais comum no mar do que na terra, devido a seu alto custo [3]. O processo de aquisição 2D é caracterizado por uma única linha de hidrofones igualmente espaçados e presos ao navio – Figura 2.6a. Já no caso da aquisição 3D, tem-se várias linhas igualmente espaçadas com um mesmo número de hidrofones em cada – Figura 2.6b.

[2] `http://www.open.edu/openlearn/science-maths-technology/science/environmental-science/earths-physical-resources-petroleum/content-section-3.2.1`, 06/11/2014



(a) Malha de sensores da aquisição sísmica 2D



(b) Malha de sensores da aquisição sísmica 3D

Figura 2.6: Aquisição Sísmica 2D e 3D no mar.

2.1.2 Processamento Sísmico

Após a aquisição do dado sísmico, o processamento aplicado tem por objetivo lapidar o dado, removendo energias indesejadas e ruído, amplificando o sinal sísmico, conduzindo os eventos as suas posições corretas, entre outros [3].

De acordo com Yilmaz [5], o processo de aquisição pode produzir dados com baixa proporção de sinal por ruído – *signal to noise ratio* –, ou seja, dados sísmicos de baixa qualidade. Muitos são os fatores que prejudicam o processo de aquisição e, por sua vez, afetam a qualidade do dado, entre eles: as condições do tempo, a presença de camadas de grande absorção de sinal, a proximidade com o ecossistema ou povoados, falhas de sensores e equipamentos de gravação, falha ao calibrar equipamento de posicionamento global, etc. Desta forma, espera-se amplificar o sinal sísmico e reduzir o ruído através do processamento para, assim, aumentar a qualidade do dado.

Contudo, o processamento não garante o aumento de qualidade do dado, principalmente porque suas etapas podem apresentar efeitos indesejados quando não são

aplicadas corretamente. Além disso, a escolha do *workflow* de processamento – conjunto de etapas de processamento – a ser aplicado, a configuração dos parâmetros das etapas e as diferentes implementações de uma mesma etapa que variam de ferramenta para ferramenta afetam o resultado final do processamento. Por essa razão, é possível obter diferentes resultados após realizar vários processamentos do mesmo dado sísmico [5].

Ainda de acordo com Yilmaz [5], o processamento deve ser cuidadosamente escolhido e executado, pois suas etapas, que são processamentos matemáticos, inserem ruídos no dado sísmico. O ruído (ou erro) pode se propagar pelo *workflow* de processamento, afetando a qualidade do dado e conduzindo os especialistas a interpretações equivocadas.

Diversos *workflows* de processamento foram propostos na literatura por diferentes autores, como por exemplo Yilmaz [2], Sheriff e Geldart [6], Bacon et al. [3], entre outros. Entre semelhanças e diferenças, os *workflows* possuem etapas de processamento em comum, mas que podem ser organizadas em diferentes estágios.

Por exemplo, Yilmaz [2] divide o *workflow* de processamento de dados sísmicos em dois estágios, a saber: pré-processamento e processamento. No estágio de pré-processamento, são realizadas etapas como a de-multiplexação, remoção de traços falhos e correções da distribuição geométrica. Já o estágio de processamento é marcado pela aplicação de etapas não menos importantes, como a deconvolução, a correção de Normal Move-out (NMO) e Dip Move-out (DMO), stack e migração.

Por outro lado, Sheriff e Geldart [6] organizam o *workflow* de processamento de dados sísmicos em três estágios diferentes, a saber: edição, processamento principal e processamento final. O estágio de edição se equipara ao estágio de pré-processamento de Yilmaz [2] em que são removidos traços falhos e é realizada a correção da distribuição geométrica. Já o estágio de processamento principal engloba etapas como deconvolução, análise e equalização de amplitudes, correção de NMO e DMO, stack, etc. Finalmente, o estágio de processamento final tem como etapa principal a migração.

Independente da organização do *workflow* de processamento, algumas etapas são cruciais e estão presentes em grande parte deles, como é o caso da deconvolução, correção de NMO e DMO, stack e migração. Destas, a deconvolução é responsável por aumentar a resolução temporal e produzir uma imagem da superfície da terra; a etapa de correção de NMO corrige o tempo de chegada da onda refletida nos sensores em função do deslocamento – *offset* – que existe entre eles; a correção de DMO é semelhante à correção de NMO, mas ocorre nos casos em que a profundidade interfere no tempo de chegada da onda refletida nos sensores; o stack é responsável por aumentar a quantidade de sinal por ruído do dado, diminuindo ou eliminando ruídos aleatórios; e a migração desloca os eventos para o seu local de ocorrência e

está associado principalmente a visualização do dado sísmico.

Destas etapas, o stack consiste em somar traços que representam um mesmo ponto da superfície para que seja possível anular o ruído aleatório e amplificar o sinal refletido. Por essa razão, ele é capaz de reduzir consideravelmente o tamanho do dado sísmico. Tamanha é sua importância que a literatura costuma distinguir duas fases do processamento dos dados sísmicos: a *pré-stack* e a *pós-stack*.

2.1.3 Interpretação Sísmica

Após a aquisição e correto processamento do dado sísmico, os geofísicos tem em mãos um dado sísmico de qualidade que reflete a estrutura geológica de uma parte da superfície da Terra. Este dado é finalmente interpretado e pode ser utilizado para determinar se existe ou não hidrocarbonetos numa região, por exemplo.

Durante a interpretação, é comum a utilização de plataformas iterativas que permitem a realização de diversas análises sobre o dado. É possível, então, visualizá-los sob várias perspectivas e direções, gerar gráficos, visualizar seções da superfícies, realizar cálculos, etc. Desta forma, o geofísico pode determinar a existência de hidrocarbonetos, identificar as estruturas geológicas que estão em volta dos reservatórios, obter uma estimativa da produção do reservatório, determinar o melhor ponto para perfuração, entre outros [5].

Portanto, a interpretação do dado está diretamente ligada a aquisição e processamento. Isso quer dizer que erros na aquisição e processamento podem levar à interpretações erradas e, consequentemente, prejuízos financeiros.

2.2 Compressão de Dados Sísmicos

Como todo *Big Data*, os dados sísmicos são volumosos e necessitam de grande poder computacional para processamento e análise. Porém, o seu volume também traz restrições ao armazenamento e transmissão, sendo comum o uso de técnicas de compressão de dados para reduzir seu tamanho. Por essa razão, a literatura é vasta e apresenta diversos trabalhos que serão organizados e descritos nesta sessão.

As técnicas de compressão de dados sísmicos podem ser divididas em dois grupos: *sem perdas* e *com perdas*. As técnicas de compressão classificadas como sem perdas são aquelas que promovem alterações reversíveis no conjunto de dados. Ou seja, um dado sísmico submetido aos processos de compressão e descompressão é idêntico ao dado sísmico anterior a estes processos.

Por outro lado, as técnicas de compressão com perdas descartam parte dos dados durante o processo de compressão, inserindo ruído e tornando o dado descomprimido diferente do dado original. Todavia, a perda causada pode ser imperceptível em

circunstâncias normais. Por exemplo, as técnicas de compressão de vídeo e áudio geralmente descartam parte dos dados e, mesmo assim, a perda é imperceptível aos olhos e ouvidos humanos, respectivamente.

Além disso, as taxas de compressão na compressão com perdas são maiores do que na compressão sem perdas. Por exemplo, Donoho et al. em [7] diz que as técnicas de compressão com perdas podem atingir taxas de compressão de 100:1, enquanto que as técnicas sem perdas atingem taxas de 2:1. Contudo, é necessário estabelecer um limite aceitável para que o ruído inserido na compressão com perdas não interfira nas etapas de processamento e interpretação do dado sísmico.

quanto às técnicas de compressão sem perdas, estas podem ser aplicadas em *amostras reestruturadas* – representadas de maneira diferente – ou em *amostras originais*. A maneira mais simples, mas não a mais eficiente, é a *compressão direta* de amostras originais, ou seja, a compressão direta do dado sísmico. Neste caso, algoritmos de compressão, como *Huffman* [8] e *Arithmetic Coding* [9], são utilizados para reduzir o tamanho dos dados.

Por outro lado, técnicas que buscam *reestruturar as amostras* do dado sísmico e em seguida comprimi-lo podem ser de três tipos: técnicas de compressão que tentam prever as amostras através de combinações lineares, subtrair a previsão das amostras reais e comprimir o resíduo da subtração; técnicas que separam sinal, expoente e mantissa antes da compressão; e técnicas que envolvem o uso de transformadas de inteiros. A semelhança entre estas técnicas é que todas procuram reduzir a entropia do dado para aumentar as taxas de compressão.

No que diz respeito às técnicas de compressão com perdas, os trabalhos relacionados exploram a aplicação de *transformadas* para reduzir a entropia do conjunto de dados. Assim, as abordagens envolvem o uso de transformadas *Cosseno*, *Wavelet* e *Lapped Orthogonal Transform* (LOT).

Nota-se que a taxonomia separa transformadas Cosseno, como *Discrete Cosine Transform* (DCT), e LOT, apesar da DCT ser um exemplo de LOT. Esta decisão foi tomada, pois os trabalhos relacionados tendem a desvincular o desempenho de LOT e transformadas cosseno. Assim, LOT pode ser considerado como um conjunto de transformadas mais genéricas que DCT.

Não obstante, destaca-se que outros tipos de transformadas também foram aplicadas na compressão de dados sísmicos, como *dreamlet* [10] e *wave packet* [11]. Todavia, o número de trabalhos que descrevem e comparam tais técnicas é bem escasso e não justificam a criação de outras classificações de transformada na taxonomia. O mesmo ocorre com técnicas que realizam a compressão com perdas, mas de outra maneira que não seja com o uso de transformadas, como por exemplo através de PCA e redes neurais [12].

Portanto, a organização da literatura pode ser resumida pela taxonomia proposta

e ilustrada na Figura 2.7. O restante desta sessão apresenta os trabalhos relacionados à compressão de dados sísmicos e estão devidamente organizados de acordo com a taxonomia proposta.

2.2.1 Compressão sem Perdas

Como dito anteriormente, a compressão sem perdas é aquela cujas alterações no dado sísmico são reversíveis e, portanto, o dado descomprimido é idêntico ao dado original – antes de ser submetido à compressão. Esta classificação é dividida em *reestruturação de amostras* e *amostras originais*.

2.2.1.1 Reestruturação de Amostras

Na *reestruturação de amostras*, as técnicas de compressão procuram reduzir a entropia do dado sísmico antes de comprimí-lo, representando suas amostras de maneira diferente. Desta forma, é possível aumentar as taxas de compressão. Esta classificação é dividida em *predição de amostras e compressão de resíduos*, *separação de sinal, expoente e mantissa*, e *transformada de inteiros*.

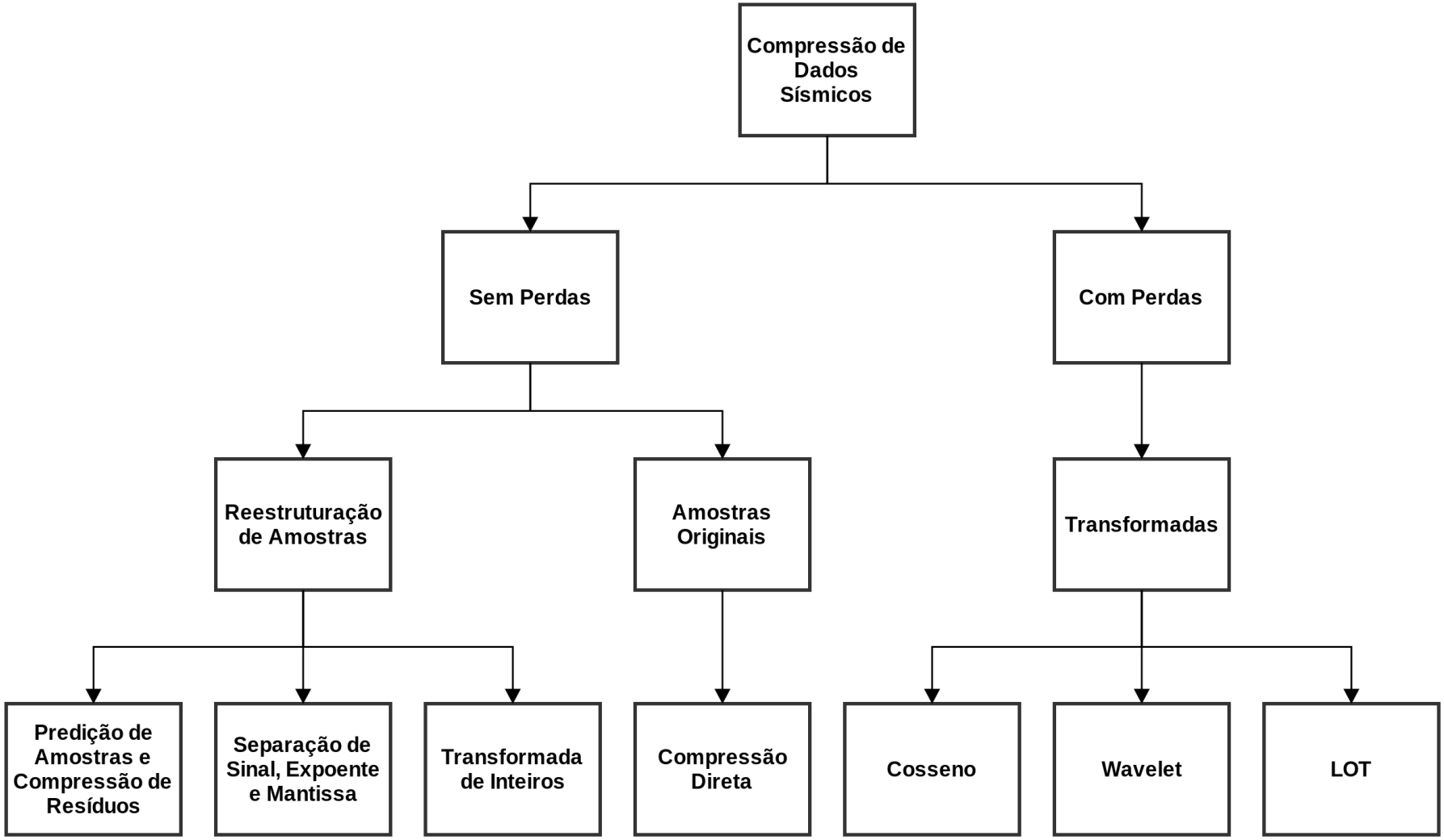


Figura 2.7: Taxonomia proposta para organizar a área de compressão de dados sísmicos

Predição de Amostras e Compressão de Resíduos

Nesta classificação, as técnicas de compressão procuram reduzir a entropia dos dados sísmicos antes da compressão. Deta forma, é possível aumentar as taxas de compressão e, consequentemente, reduzir o tamanho do arquivo comprimido.

A predição linear é uma técnica de compressão de dados sísmicos proposta por Stearns et al. em [13, 14] que possui duas fases: descorrelação e codificação. O objetivo da primeira fase é explorar as redundâncias das amostras, descorrelacionado-as e reduzindo o número de bits necessários a sua representação [15]. Nesta fase, os dados são representados de maneira diferente, mas nenhuma compressão é realizada. Já na segunda fase, algoritmos de compressão, como *Bi-Level Coding* e *Arithmetic Coding*, são aplicados para reduzir o tamanho do dado, sempre que possível.

Originalmente, esta técnica era conhecida por LPBLC – *Linear Prediction with Bi-Level Coding* – e empregava *Bi-Level Coding* na compressão dos resíduos. Contudo, estudos posteriores apontaram que uma adaptação no algoritmo *Arithmetic Coding* permite que este alcance melhor performance na compressão de dados sísmicos do que o *Bi-Level Coding* [16].

Conforme pode ser visto na Figura 2.8, uma sequência de amostras de dados sísmicos – $x(0 : K-1)$ – são submetidas à predição linear. O processo consiste em utilizar as M primeiras amostras de $x(0 : K-1)$ para prever as $K-M$ amostras subsequentes, ou seja, usa-se $x(0 : M-1)$ para prever $x(M : K-1)$. Para tal, usa-se as M amostras originais e os coeficientes da predição – $b(0 : M-1)$ – numa combinação linear que determina a previsão das amostras subsequentes. A sequência predita – $x'(M : K-1)$ – é, então, subtraída da sequência original – $x(M : K-1)$ – produzindo o resíduo – $r(M : K-1)$. Por fim, comprime-se as M amostras utilizadas na predição, os coeficientes da predição $b(0 : M-1)$ e os resíduos.

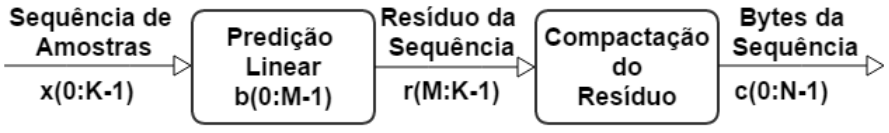


Figura 2.8: Arquitetura de compressão da predição linear.

A principal vantagem em comprimir os resíduos e não as amostras originais é que aqueles possuem menor entropia do que estas e, consequentemente, obtém-se maiores taxas de compressão. Ademais, é possível melhorar a eficiência da predição ao aumentar a quantidade M de amostras utilizadas no processo. Apesar desta medida diminuir ainda mais a entropia dos resíduos, também se aumenta a quantidade de amostras originais que precisam ser comprimidas diretamente – o que pode diminuir a taxa de compressão. Ou seja, é preciso encontrar um valor ótimo para M tal que

a predição linear produza as maiores taxas de compressão.

Quanto aos aspectos negativos, cita-se o fato de que os coeficientes da predição precisam ser calculados dinamicamente [17], o que aumenta o *overhead* do método. Além disso, para sequências de amostras em ponto flutuante é preciso aplicar as mesmas operações de truncamento e arredondamento para garantir que a compressão seja de fato sem perdas [18]. Por fim, esta técnica exige que os coeficientes da predição sejam comprimidos ou transmitidos juntamente com os resíduos, enquanto outras técnicas exigem a compressão ou envio apenas dos resíduos e amostras.

Com relação aos trabalhos relacionados, Honea e Stearns em [19] avaliaram a predição linear variando seus parâmetros. Em uma configuração, a predição linear utiliza *frames* de 1000 e 4000 amostras e número de coeficientes dinamicamente calculados para cada *frame*. Em outro momento, opta-se por *frames* de 4000 amostras e número de coeficientes fixados em nove. Já na fase de codificação, utilizou-se tanto *Bi-Level Coding* quanto *Arithmetic Coding*.

Como conclusões, os autores observaram que quanto maior o tamanho do *frame*, melhor é a compressão e maior é o tempo de execução. Além disso, a fixação do número de coeficientes da predição em nove reduziu a complexidade do algoritmo e, consequentemente, o tempo de execução. Quanto à fase de codificação, o algoritmo *Arithmetic Coding* obteve melhor desempenho na compressão dos dados sísmicos do que o algoritmo *Bi-Level Coding*.

Também buscando melhorar as taxas de compressão do LPBLC, Fan et al. em [20] propõe algumas melhorias: busca-se otimizar o cálculo dos parâmetros do *Bi-Level Coding*, calcular o tamanho do frame na predição linear dinamicamente e otimizar a configuração dos seus parâmetros – como número de coeficientes. Com estas alterações, é possível obter melhor desempenho na compressão de dados sísmicos.

Uma alternativa ao LPBLC apresentada por McCoy et al. em [18] propõe o uso de filtros adaptativos, uma vez que este é mais eficiente em tempo de execução do que a predição linear "tradicional". De maneira semelhante, Mandyam et al. em [15] apresenta a formulação matemática de três filtros adaptativos: *Normalized Least Mean Square* (NLMS), *Gradient Adaptive Lattice* (GAL) e *Recursive Least Square Lattice* (RLSL). Destes, o RLSL é o melhor em taxa de compressão, apesar de ser o de maior complexidade computacional.

Outra alternativa ao uso de LPBLC foi proposta por Peterson e Hutt em [21] em que amostras consecutivas eram subtraídas e o resíduo comprimido com *Steim1* e *Steim2*. Os resultados mostraram que a técnica LPBLC obtém maiores taxas de compressão do que a diferença entre amostras consecutivas associada a estes codificadores – comprimindo duas vezes mais, no melhor caso. Contudo, o esquema proposto pelos autores é cerca de 50 vezes mais rápido do que o LPBLC.

Peterson e Hutt pregam que a diferença entre amostras consecutivas é uma boa

opção a ser aplicada em dados sísmicos com pequena variação de frequência. Como as amostras destes dados estão próximas em magnitude, a diferença entre elas produz resíduos próximos de zero com entropia tão baixa quanto à entropia dos resíduos produzidos pela predição linear. Além disso, as técnicas de codificação *Steim1* e *Steim2* são utilizadas pela *U.S. Geological Survey*[3] (USGS).

Nijim et al. em [22] expandiu a aplicação da diferença entre amostras consecutivas, propondo três esquemas para reduzir a entropia das amostras do dado sísmico. Primeiro, os autores propuseram subtrair amostras consecutivas – diferença de primeira ordem –, ou subtrair o resultado da subtração – diferença de segunda ordem –, e assim por diante. A segunda proposta consiste em calcular o resíduo combinando diferentes ordens de subtração, tornando possível reduzir ainda mais a variação do resíduo. Finalmente, a terceira e última proposta consiste em utilizar séries de Taylor para aproximar as amostras. Já na fase de codificação, opta-se pelo algoritmo *Arithmetic Coding*.

Os autores argumentam que, além de mais rápida, esta abordagem é direta e produz taxas de compressão comparáveis ao LPBLC. Além disso, não é preciso transmitir ou comprimir os coeficientes da predição linear, visto que os coeficientes utilizados na diferença entre amostras não são determinado dinamicamente.

Fargues et al. em [23] combina a diferença de quarta ordem entre amostras com *adaptive differential pulse code modulation* (ADPCM) na fase de descorrelação. Segundo os autores, o uso da diferença e não de filtros adaptativos permite que o esquema de compressão seja executado em tempo real.

O maior problema da diferença entre amostras é que esta não possui um bom desempenho em dados sísmicos que possuem muito ruído ou cuja variação de frequência é muito alta. Neste caso, o resíduo da diferença é comprimido tão bem quanto às amostras.

Outra opção para ser utilizada na fase de descorrelação é uma abordagem conhecida como Pole-Zero [24, 25] que opera no domínio da frequência e minimiza o erro quadrático médio. A abordagem utiliza funções de polos e zeros para modelar as amostras originais no domínio da frequência. Desta forma, as amostras originais e estimadas são subtraídas, originando o resíduo que é comprimido com *Arithmetic Coding*. A performance deste método se equipara à predição linear e é melhor do que a diferença entre amostras consecutivas. Contudo, a técnica foi desenvolvida para dados símicos cujas amostras são números inteiros e, portanto, causa perda de precisão para amostras de ponto flutuante.

Abanmi et al. em [26] propõe uma predição linear diferente das descritas até o momento. Segundo eles, é possível aproximar as amostras dos dados símicos através de uma soma finita de sinusoides complexas e comprimir o resíduo com *Bi-Level*

[3] `http://www.usgs.gov`

Coding. Está técnica não foi comparada com outras técnicas presentes literatura.

Separação de Sinal, Expoente e Mantissa

Esta classificação apresenta técnicas de compressão que separam as amostras sísmicas em três partes – sinal, expoente e mantissa – e as comprime independentemente. Além disso, os dados sísmicos submetidos a este tipo de compressão são formados por amostras de 32 bits em ponto flutuante.

O único trabalho encontrado nesta categoria foi apresentado por Xie e Qin em [27]. Eles descrevem uma abordagem para comprimir arquivos SEG-Y, desconsiderando os cabeçalhos e comprimindo apenas as amostras.

Este esquema de compressão propõe utilizar a diferença de amostras para reduzir a entropia do dado sísmico, obtendo, assim, os resíduos. Em seguida, tanto os resíduos quanto às amostras originais são separadas em sinal, expoente e mantissa, e comparados. Caso o sinal seja o mesmo para resíduos e amostras, e os expoentes satisfaçam os critérios estabelecidos, então o sinal, expoente e mantissa dos resíduos são comprimido. Caso contrário, comprime-se o sinal, expoente e mantissa das amostras. Como codificador, utiliza-se *Arithmetic Coding*.

Apesar de ser uma abordagem destinada à compressão de arquivos SEG-Y, apenas as amostras são comprimidas, enquanto os cabeçalhos são desprezados. Portanto, é possível melhorar o desempenho ao comprimir também os cabeçalhos.

Ademais, o uso da diferença como técnica de descorrelação não traz garantias de que a entropia do dado será reduzida. Na realidade, para determinar os coeficientes e o número de amostras que serão utilizadas na diferença, é necessário conhecer previamente o dado sísmico. Desta forma, a técnica de descorrelação apresenta desempenho variável e dependente do dado sísmico.

Além disso, o esquema de compressão precisa decidir entre comprimir amostras ou resíduos, o que aumenta o *overhead* da compressão e descompressão.

Transformada de Inteiros

Nesta classificação, as técnicas de compressão possuem duas fases. Na primeira fase, uma transformada é aplicada com o objetivo de reduzir a entropia dos dados sísmicos, representando as amostras por coeficientes da transformada. Portanto, ao aplicá-la, as amostras que estão no domínio do espaço são representadas em outro domínio, como o domínio da frequência. Contudo, nenhuma compressão é realizada e o dado sísmico possui o mesmo tamanho, apesar da diferente representação. A redução no tamanho do dado sísmico ocorre apenas na segunda fase em que um algoritmo de compressão é aplicado.

Todavia, para que seja possível realizar a compressão sem perdas, é preciso utilizar uma transformada que produza coeficientes inteiros, como a transformada *Integer Wavelet*. Caso os coeficientes da transformada sejam números em ponto flutuante, então o sinal reconstruído após o processo de descompressão não é idêntico ao sinal original.

Dos trabalhos encontrados na literatura, todos utilizam a transformada *Integer Wavelet* implementada no esquema *lifting* [28]. Este esquema é responsável por reduzir a complexidade do algoritmo da transformada *Wavelet*, produzir coeficientes inteiros através de operações de arredondamento e é composto por três etapas: *split*, *prediction* e *update*.

Na primeira etapa, o sinal original é separado em dois conjuntos, um com amostras pares e outros com amostras ímpares – *even* e *odd*, respectivamente. Na segunda etapa, as amostras pares são utilizadas para prever as amostras ímpares e esta previsão é subtraída das amostras ímpares, resultando em resíduos – *prediction error*. Na última etapa, o resíduo é somado às amostras pares para melhorar a previsão na próxima rodada. Portanto, a etapa *prediction* – ou *dual lifting* – é responsável por calcular os coeficientes da transformada *Wavelet* (γ_j), enquanto que a etapa *update* – ou *primal lifting* – é encarregada de calcular os coeficientes de escala (λ_j), mas ambas possuem operações de arredondamento em sua composição, o que garante a geração de coeficientes inteiros [29]. A Figura 2.9 resume este processo.

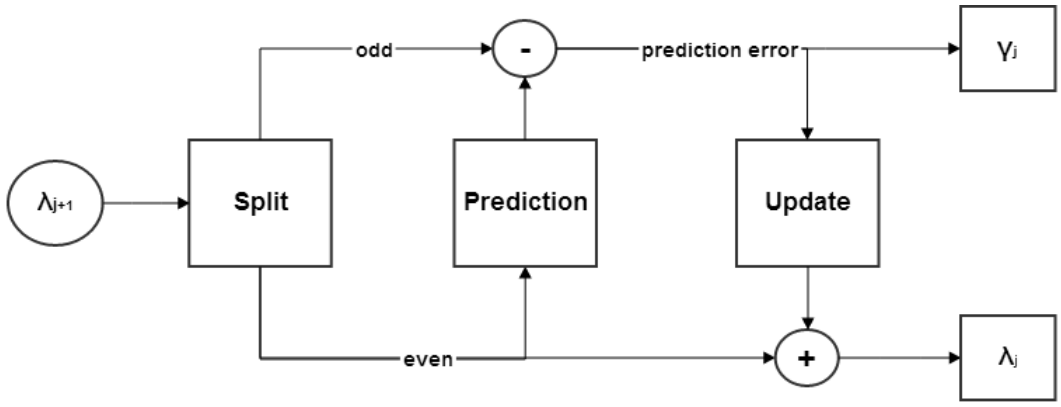


Figura 2.9: O esquema *lifting*.

Uma das vantagens em se utilizar o schema *lifting* é que ele é reversível e, portanto, permite que os dados originais sejam recuperados sem perdas. Ademais, ele não requer a utilização de vetores temporários durante os cálculos, permitindo um baixo consumo de memória.

Quanto aos trabalhos relacionados, Wang et al. em [29] comparam o desempenho de diferentes funções *Wavelet* – *Cohen-Daubechies-Feauveau* (CDF), *SWE(13,7)* e *CRF(13,7)*, sendo as duas últimas utilizadas no JPEG2000 – e variam o tamanho do filtro da transformada. Os primeiros resultados mostraram que a aplicação da

transformada *Integer Wavelet* e posterior codificação apresenta maior taxa de compressão do que algoritmos de compressão *Huffman* e *LZW* aplicados diretamente no dado sísmico. Além disso, entre as transformadas e filtros experimentados, a transformada *Wavelet* de melhor desempenho é a *CDF(2,n)*.

Zheng e Liu em [30] apresentam um esquema de compressão paralela para dados sísmicos do tipo *RAW*, que é composto por 512 bytes de cabeçalhos e 4 canais de amostras. Neste esquema, a parte opcional do header é descartada e as amostras são divididas em quatro vetores que são comprimidos em paralelo. A compressão das amostras envolve o uso da transformada *Integer Wavelet* com decomposição de dois níveis que gera coeficientes de frequências alta e baixa, sendo comprimidos em seguida. O esquema de compressão paralela foi capaz de reduzir o tempo de execução em 30%, comparando-se com as abordagens tradicionais.

Por fim, Meftah et al em [31] propõe uma abordagem para otimizar o cálculo dos coeficientes da transformada *Integer Wavelet* e reduzir a entropia dos coeficientes de baixa frequência. Ao invés de utilizar um filtro clássico, vertical ou horizontal, os autores propõem um filtro que segue os "horizontes". Os "horizontes" são superfícies que representam diferentes camadas rochosas da Terra e sua direção varia de acordo com atributos do dado sísmico, mas dificilmente são verticais e horizontais. Adotando esta estratégia é possível reduzir o tamanho do arquivo comprimido em até 8%.

Um das dificuldades em se usar a transformada *Integer Wavelet* é que ela requer que o tamanho do sinal de entrada seja uma potência de dois – por exemplo, 2, 4, 8, 16,... , 1024, etc – e não há nenhuma garantia que a quantidade de amostras em um traço de dado sísmico satisfaça essa característica. Portanto, é necessário adaptar os algoritmos de compressão para atender a esta necessidade.

Uma modificação simples é incluir zeros nos traços do dado sísmico até que o tamanho do sinal seja um múltiplo de potência de dois. Contudo, exige-se mais memória e poder computacional com esta solução, principalmente quando existem muitas amostras por traço. Por outro lado, é possível dividir o traço em pequenos blocos de amostras cujo tamanho é um múltiplo de potência de dois e completar com zeros o último bloco. Entretanto, esta opção aumenta a complexidade do algoritmo e pode impactar negativamente nas taxas de compressão.

2.2.1.2 Amostras Originais

Nesta classificação, as técnicas de compressão são aplicadas diretamente nos dados sísmicos. Ou seja, as amostras do dado sísmico são comprimidas da maneira em que estão representadas, sem nenhum tipo de pré-processamento adicional.

Compressão Direta

Apenas duas técnicas de compressão foram aplicadas diretamente a dados sísmicos e não apresentam um bom desempenho, visto que suas amostras podem apresentar grande variação. O trabalho que descreve a aplicação foi desenvolvido por Ahmed Adnan Aqrawi na Universidade Norueguesa de Ciência e Tecnologia para obtenção do título de mestre [32].

Na sua dissertação, Aqrawi compara o desempenho de várias técnicas de compressão de dados sísmicos, tanto com perdas quanto sem perdas. No que diz respeito à compressão sem perdas, Aqrawi implementa o algoritmo de *Huffman* em três versões: sequencial, paralela em CPU e paralela em GPU. Ao compará-las, observa-se quem todas alcançam *compression ratio* de 1.4, porém a versão paralela em CPU é mais veloz que as demais versões, enquanto que a versão paralela em GPU é mais rápida que a versão sequencial.

Mesmo que a GPU tenha capacidade de executar mais threads simultâneas do que o processador i7 utilizado nos experimentos, a versão paralela em GPU não é mais rápida do que a versão paralela em CPU. Isso ocorre porque a CPU é mais eficiente em operações de bits do que a GPU. Além disso, destaca-se que os algoritmos de compressão sem perdas são difíceis de serem paralelizados devido a sua natureza sequencial. Desta forma, muitas implementações são cheias de blocos *if-else* que tornam a execução mais lenta.

No segundo experimento, Aqrawi sugere a compressão do dado sísmico utilizando o algoritmo *Run-Length Encoding* (RLE). Contudo, ele argumenta que aplicar esta técnica de compressão a nível de bits pode produzir arquivos comprimidos de tamanho maior do que o arquivo original, ou seja, não haveria compressão. Isso porque os dados sísmicos são formados por amostras variáveis e ruído de baixa compressão, tornando difícil encontrar longas sequências de bits iguais.

Desta forma, ele sugere uma implementação de RLE otimizado a nível de byte, buscando comprimir apenas sequências repetidas do byte zero. Além disso, um bit foi acrescentado no dado comprimido para determinar se o próximo valor da sequência representa o número de bytes zero que foram codificados ou se aquela sequência é uma parte do dado não comprimido.

Apenas duas versões do RLE foram criadas – uma sequencial e uma paralela em CPU – e, de maneira semalhante aos resultados anteriores, a versão paralela foi mais eficiente em tempo de compressão do que a versão sequencial.

Outro trabalho desenvolvido por Aqrawi e Elster busca medir o *speed-up* de diferentes técnicas de compressão de dados sísmicos com e sem perdas, variando o hardware utilizado nos experimentos – os autores propuseram dois sistemas diferentes com CPU, GPU e discos diferentes, no caso dos discos, comparou-se HDDs e SSD.

Nota-se, portanto, uma preocupação clara com o tempo de execução dos algoritmos, visto que em muitas aplicações sísmicas o tempo é uma restrição. Além disso, eles propuseram dois modelos para prever o tempo de compressão, um síncrono e outro assíncrono [33].

quanto às técnicas de compressão sem perdas empregadas, foram as mesmas descritas nesta sessão: *Huffman* e RLE. Elas foram escolhidas por serem rápidas, apesar de apresentarem um desempenho não tão bom na compressão de dados sísmicos. Os resultados mostraram que em discos HDDs, o *speed-up* dos algoritmos ficou próximo 1.1 para o modelo síncrono e 1.3 para o modelo assíncrono. Todavia, os experimentos executados com SSD apresentaram *speed-up* negativo, mostrando que os algoritmos são lentos ou a compressão é muito pequena.

2.2.2 Compressão com Perdas

Nesta classificação, as técnicas de compressão descartam parte do dado sísmico durante sua compressão. Por ser um processo irreversível, a descompressão não é capaz de reconstruir o dado exatamente como era antes da compressão. Entretanto, a perda causada não é perda de traços ou amostras, mas sim a perda de precisão nas casas decimais das amostras do dado sísmico.

Na realidade, estas técnicas quando aplicadas de maneira controlada inserem ruídos aleatórios (ou gaussianos) no dado sísmico, sem nenhuma correlação e até imperceptíveis aos olhos humanos. Considerando que o próprio processo de aquisição insere ruído no dado sísmico e que o processamento do dado filtra o ruído, a compressão com perdas pode sim ser aceitável como foi mostrado por Averbuch et al. em [34] e Rosten e Amundsen em [35]. Mesmo assim, estas técnicas não são bem aceitas por geofísicos e geólogos.

Devido ao descarte de parte dos dados, a compressão com perdas é capaz de atingir taxas de compressão muito mais elevadas do que a compressão sem perdas. Contudo, quanto maior a taxa de compressão, maior é a perda, o que pode tornar o método inaceitável. Desta forma, é preciso controlar as taxas de compressão para que o ruído inserido mantenha as propriedades de um ruído gaussiano e sem ultrapassar a quantidade de ruído inserido no processo de aquisição [36].

Não existe um valor máximo para a taxa de compressão com perdas de maneira a evitar danos no dado sísmico. Estas variam com o dado, o que implica que uma taxa pode ser aceitável para um dado e inaceitável para outro. O ideal é que as taxas de compressão sejam adaptativas [36] e que as técnicas de compressão incorporem um controle de qualidade que garantam a integridade do dado sísmico [37].

Todavia, não é fácil estabelecer métricas de qualidade para a compressão de dados sísmicos e determinar o quanto de perda é aceitável. Muitos autores procu-

ram detectar artefatos visuais, comparando visualmente o dado sísmico original e o descomprimido. Contudo, esta prática não é eficaz, pois mesmo que as diferenças não sejam visíveis, o erro (medido por MSE, por exemplo) pode ser relativamente grande. Outros autores preferem utilizar a taxa de sinal por ruído, mas esta métrica não reflete a correlação do ruído no dado sísmico que é um fator a ser considerado.

Além disso, a aplicação do dado sísmico está diretamente ligado ao nível de perda aceitável. Por exemplo, quando os dados são utilizados para visualização, é possível aplicar elevadas taxas de compressão, desde que não prejudiquem as características geofísicas visíveis. Por outro lado, quando o dado sísmico é utilizado em *workflows* de processamento, a compressão com perdas é aceitável principalmente se o *workflow* contiver etapas de atenuação de ruído, como *stack* e migração. Porém, etapas como a deconvolução podem amplificar o erro inserido na compressão e comprometer o dado sísmico.

2.2.2.1 Transformadas

A técnica mais comum empregada na compressão com perdas de dados sísmicos envolve o uso de transformadas. Geralmente, estas técnicas são divididas em três etapas: transformação, quantização e codificação [38]. Na etapa de transformação, uma transformada, como *Discrete Wavelet* [39] ou *Discrete Cosine* [40], é aplicada para reduzir a entropia do dado. Desta forma, as amostras do dado sísmico que estão representadas no domínio do espaço são representadas em outros domínios, como tempo e frequência, na forma de coeficientes da transformada.

Na etapa da quantização, o conjunto de coeficientes da transformada é mapeado em um conjunto menor de valores inteiros, reduzindo a entropia do dado, mas sem comprimi-lo. Existem dois tipos de quantização: escalar e vetorial. A diferença entre elas é que a quantização escalar opera sobre valores individuais, arredondado-os por exemplo, enquanto que a quantização vetorial atual sobre um vetor de coeficientes simultaneamente. Como este processo mapeia um conjunto de valores em um único valor, ele é irreversível e está diretamente ligado a compressão com perdas – note que o uso da transformada na compressão sem perdas não apresenta o passo da quantização.

Após estes passos, o dado sísmico pode ter seu tamanho reduzido significativamente, já que sua nova representação possui menor entropia que a representação no domínio do espaço. Desta forma, a etapa da codificação emprega um algoritmo de compressão que reduz o tamanho do dado sísmico.

Cosseno

Discrete Cosine Transform (DCT) aproxima o sinal por uma soma de funções cosseno de diferentes frequências, representado o dado sísmico no domínio da frequência. Desta forma, é possível reter grande parte da informação em poucos coeficientes de baixa frequência, enquanto que a maior parte dos coeficientes de alta frequência pode ser descartado. Dentre as muitas aplicações de DCT, destaca-se o uso na compressão de sinais e imagens, sobretudo em JPEG. Talvez, por esta razão, DCT foi empregada na compressão de dados sísmicos.

Dos trabalhos relacionados, Spanias et al. em [41] comparam a performance de quatro transformadas – *Discrete Fourier Transform* (DFT), DCT, *Walsh-Hadamard Transform* (WHT) e *Karhunen-Loeve Transform* (KLT) – na compressão de dados sísmicos. A compressão utilizando as transformadas foram implementadas em dois modos diferentes: uniforme e sub-banda. No primeiro, o número de bits por faixa de frequência é uniforme. Já no segundo, o número de bits em diferentes faixas frequência varia e é maior em faixas de alta frequência. Além disso, foram testados dois esquemas de quantização um uniforme e outro não uniforme.

Os resultados mostraram que a implementação da transformada em sub-banda e com quantização uniforme é mais eficaz do que as demais. Quanto ao desempenho das transformadas, apesar da KLT alcançar melhor resultado na compressão de dados sísmicos, ela não é robusta, ou seja, seu desempenho depende do dado. Por essa razão, os autores elegem a DCT, dentre as experimentadas, como a transformada ideal para compressão de dados sísmicos.

Vermeer et al. em [42] defendem o uso de compressão com perdas para dados sísmicos, que deve ser encarada como mais um processo gerador de ruído. O ruído gerado pela compressão deve ser aleatório, como o ruído do ambiente, que constitui um limite natural para o ruído da compressão. Os autores ainda realizam alguns experimentos em dados sísmicos, comprimindo-os com DCT e medindo a quantidade de ruído inserido no processo.

Bernasconi e Vassallo em [43] propõe uma abordagem totalmente diferente para comprimir dados sísmicos obtidos a partir da perfuração de poços. Neste caso, o dado sísmico é convertido para o domínio angular, em que os eventos sísmicos possuem maior correlação do que no domínio espacial. Em seguida, aplicada-se a transformada DCT 2D e elimina-se coeficientes via *threshold*. Esse método foi comparado com Transformada Wavelet 1D e alcançou melhores taxas de compressão e qualidade de imagem descomprimida.

Meyer em [44] detalha a implementação de um esquema de compressão de transformada que utiliza a *Local Cosine Transform* (LCT), uma vez que esta é mais recomendada para dados com alta frequência do que a transformada *Wavelet*. Sua descrição é bem detalhada e vai desde as escolha de parâmetros da transformada até a escolha do quantizador. Na fase de codificação, utiliza-se *Arithmetic Coding* e seus

experimentos são realizados sobre duas imagens de dados sísmicos, uma sintética e a outra real.

Como a localização das bases da LCT utiliza busca binária, Wang e Wu em [45] propõem uma melhoria para cortar nós da árvore – *pruning* – que possuem entropia maior do que a entropia do nó pai, já que não constituem a melhor base trigonométrica. Desta forma, surgiu a *Adaptive Local Cosine Transform* (ALCT) que pode ser de dois tipos: *time-ALCT* que é adaptativa no tempo e uniforme no espaço, e *space-ALCT* que é adaptativa no espaço e uniforme no tempo. Os resultados dos experimentos mostraram que ALCT permite maiores taxas de compressão e imagens de maior resolução do que a LCT.

Uma nova melhoria foi proposta por Wu e Wang [46], mas desta vez para substituir a busca binária na LCT. A alternativa propõe que o sinal seja dividido em células menores de mesmo tamanho, como pequenos blocos de dados. Os blocos vizinhos podem ser unificados, desde que a entropia dos blocos juntos seja menor do que a soma de entropias dos blocos separados. Este algoritmo ficou conhecido como *FLEXIBLE* e apresenta quatro vantagens: não possui restrições da busca binária que podia dividir o sinal em um ponto impróprio – por exemplo, no meio de um "pico" –, é menos sensível as variações no tempo e espaço, apresenta menor distorção na reconstrução do sinal e apresenta maior desempenho na compressão de dados sísmicos, como comprovado em [47].

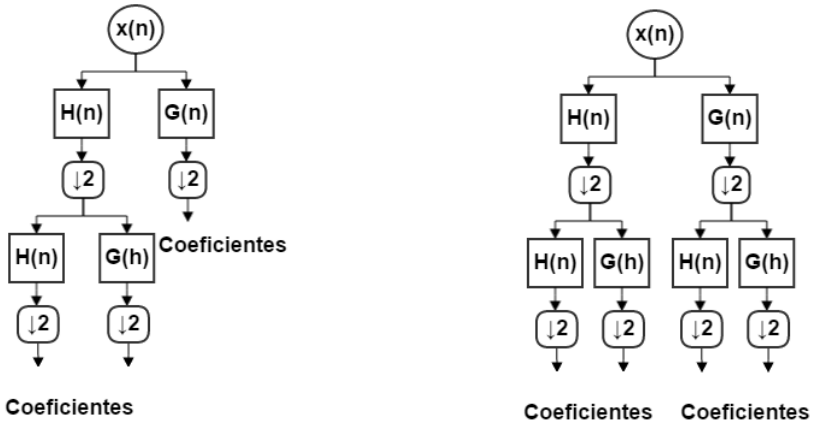
Apesar dos inúmeros estudos que envolvem a compressão de dados sísmicos utilizando a transformada cosseno, alguns autores criticam o uso da mesma. Eles argumentam que a reconstrução do dado comprimido pode gerar "artefato de blocos", como na compressão JPEG.

Wavelet

A transformada *Wavelet* se caracteriza pela aplicação de filtros de passa-alta e passa-baixa para geração de coeficientes de detalhe e aproximação, respectivamente. Os coeficientes de aproximação armazenam grande parte da informação significativa do dado sísmico, enquanto que os coeficientes de detalhe são volumosos e grande parte representa os ruídos. Por essa razão, alguns esquemas de compressão priorizam coeficientes de aproximação e até mesmo descartam os coeficientes de detalhe.

Das transformadas *Wavelet* utilizadas, destacam-se dois tipos que diferem quanto à decomposição realizada: *Discrete Wavelet Transform* (DWT) e *Discrete Wavelet Packet Transform* (DWPT). Na DWT, apenas os coeficientes de aproximação são utilizados como entrada da transformada durante a decomposição do sinal. Por outro lado, DWPT utiliza como entrada tanto coeficientes de aproximação quanto coeficientes de detalhe, gerando uma árvore binária completa e proporcionando mai-

ores taxas de compressão e menor ruído. Esta diferença é ilustrada na Figura 2.10.



(a) Discrete Wavelet Transform. (b) Discrete Wavelet Packet Transform.

Figura 2.10: Nesta figura, $x(n)$ representa o sinal sísmico, $H(n)$ o filtro passa-baixa, $G(n)$ o filtro passa-alta e $\downarrow 2$ a subamostragem por 2. Em (a), observa-se que a decomposição utilizando DWT é realizada apenas nos coeficientes de aproximação gerados por $H(n)$. Enquanto que em (b), a decomposição utilizando DWPT é realizada em coeficientes de aproximação e detalhe (gerados por $H(n)$ e $G(n)$,respectivamente), construindo uma árvore binária completa.

Independente da transformada utilizada, a compressão com perdas que as utiliza produz coeficientes em ponto flutuante e emprega a quantização ou eliminação de coeficientes via *threshold*. Desta forma, o dado sísmico descomprimido se assemelha ao dado sísmico anterior à compressão, mas não é idêntico a ele [30].

No que tange a aplicação de DWT, Bosman e Reiter em [48] verificam se a quantidade de ruído inserido na compressão com transformada *Wavelet* é aceitável e se este ruído interfere nas etapas de deconvolução, *stack* e migração. Os resultados mostraram que dados sísmicos submetidos a taxa de compressão de 25:1 são quase idênticos (visivelmente) aos dados anteriores à compressão, enquanto que a taxa de compressão de 50:1 introduziu ruídos mais visíveis nos dados. Além disso, as etapas de *stack* e migração atenuaram o efeito do ruído, enquanto que a deconvolução o ampliou.

Donoho e Villasenor em [7] defendem o uso de transformada *Wavelet*, argumentando que ela é capaz de separar o dado sísmico em grupos de informação significativa e ruído. Desta forma, o primeiro grupo deve dispor de mais bits na quantização, enquanto que o segundo grupo pode ter seus coeficientes quantizados para zero ou para números bem pequenos. Aplicando este esquema, os autores mostram que é possível atingir taxas de compressão de 100:1 sem afetar significativamente as características geofísicas.

Reiter em [49] compara e mede o desempenho de transformadas *Wavelet* de uma, duas e três dimensões na compressão de dados sísmicos, utilizando duas métricas de qualidade que refletem a quantidade de ruído inserido, mas não medem a correlação

deste. Seus experimentos mostraram que quanto maior a dimensão da transformada, menor é o ruído inserido no dado descomprimido, sobretudo para taxas de compressão mais elevadas. Além disso, o autor mostra que o ruído diminui com os processo de correção de NMO e *stack*.

Vassiliou e Wickerhauser em [50] comparam o desempenho de algoritmos de compressão de imagens baseados em DWT aplicados em dados sísmicos e argumentam que este tipo de dado apresenta descontinuidades e grande variação de amplitude, o que dificulta a compressão. Nos experimentos, variam-se o tamanho do filtro da transformada, o nível da decomposição e o tipo de quantização – totalizando 36 configurações diferentes –, e emprega-se RLE e *Huffman* na compressão. Os melhores resultados na compressão foram alcançados com filtros grandes, decomposição moderada e quantização escalar.

Khene e Abdul-Jauwad em [51] propõem a aplicação de DWT de uma dimensão nas linhas e colunas de uma matriz de amostras de dado sísmico para obter uma decomposição 2D. Para tal, aplica-se a transformada nas linhas da matriz de amostras e, posteriormente, nas colunas da matriz (de coeficientes gerados no passo anterior). Este processo é iterativo e pode ser repetido até nível da decomposição desejado. Os autores ainda utilizam um método estatístico para determinar o *threshold* e eliminar parte dos coeficientes gerados, tornando possível descartar 82% dos coeficientes e reconstruir 95% do dado sísmico.

Averbuch et al. em [34] comparam oito esquemas de compressão, variando transformadas, quantização e codificação. Dentre as transformadas utilizadas, destaca-se o uso de *Fast Wavelet Transform* (FWT), *Fast Wavelet Packet Transform* (FWPT) e *Local Cosine Transform*. Os resultados mostraram que FWT associada a quantização uniforme é no mínimo 2.5 vezes mais rápido do que os demais esquemas de compressão. Além disso, taxas de compressão moderadas inserem ruídos descorrelacionados que não afetam o processamento e interpretação do dado sísmico.

Quanto aos esquemas de compressão que utilizam DWPT, Al-Moohimeed em [52] propõe um esquema adaptativo que seleciona a melhor árvore de decomposição do sinal sísmico, considerando a entropia como critério de seleção. Além disso, ele determina o *threshold* da quantização, balanceando a quantidade de energia retida no sinal comprimido. Seus resultados mostraram que é possível atingir maior taxa de compressão do que se fossem utilizadas DWT e DCT, para uma taxa de sinal por ruído fixa, sem gerar artefatos de bloco no dado sísmico.

De maneira semelhante, Wenbo et al. em [53] aplica o esquema adaptativo para seleção da árvore de decomposição via DWPT e *threshold* da quantização, associando à compressão por *Arithmetic Coding*. Com ele, foi possível atingir taxas de compressão de 30:1, inserindo menos ruído do que com a aplicação de DWT.

LOT

Lapped Orthogonal Transform (LOT) é uma extensão de DCT que suaviza o efeito de artefatos de blocos apresentados em grandes taxas de compressão. Estas transformadas podem alcançar maiores taxa de compressão e taxa de sinal por ruído do que as transformadas *Wavelet*, apresentando melhor desempenho na compressão de dados sísmicos. Dos trabalhos relacionados, destaca-se o uso de duas transformadas: *Generalized Lapped Orthogonal Transform* (GenLOT) e *Generalized Unequal Length Lapped Orthogonal Transform* (GULLOT).

quanto à aplicação de GenLOT, Duval et al. em [54, 55] comparam o desempenho de diferentes variações de GenLOT (incluindo DCT) com a transformada *Wavelet* na compressão de dados sísmicos. Seu esquema de compressão envolve o uso da transformada (*Wavelet* ou GenLOT) e a aplicação de *Zerotree Coding*. Os resultados mostraram que é importante variar a configuração de GenLOT que são aplicadas nas diferentes dimensões do dado, tornando possível obter maior taxa de sinal por ruído do que aquela proporcionada pela transformada *Wavelet*. Além disso, quando a taxa de compressão é muito elevada, a transformada *Wavelet* insere ruídos correlacionado, enquanto que GenLOT não.

Não obstante, Duval et al. em [56] mostra que o desempenho de GenLOT associadas com *Embedded Zerotree Wavelet*, que é um algoritmo de compressão de imagens, é melhor do que os algoritmos mais eficientes de Partição de Conjuntos em Árvore Hierárquicas – *Set Partitioning in Hierarchical Trees* (SPIHT).

Também é possível otimizar GenLOT utilizando alguns critérios como *coding gain*, *stopband attenuation* e *DC leakage*, melhorando a qualidade visual do dado descomprimido. Mas nem sempre é possível alcançar melhoras significativas em qualquer taxa de compressão [57].

Quanto ao uso de GULLOT, Duval e Nagai em [58] ressaltam que o desempenho desta é similar ao desempenho de GenLOT na compressão de dados sísmicos cujo sinal é suave. Todavia, sinais sísmicos de maior frequência são melhor comprimidos por GULLOT, que também pode ser otimizada.

Todavia, ressalta-se que o design de GenLOT e GULLOT não é uma tarefa fácil, pois envolve desde a seleção dos parâmetros da transformada até a implementação das otimizações. Além disso, as otimizações podem não surtir o efeito desejado em qualquer taxa de compressão, ou seja, muitas vezes as melhorias são notáveis em alguns níveis de compressão, mas não em outros. Por fim, destaca-se que estas transformada proporcionam maior quantidade de sinal por ruído do que aquela alcançada com o uso da transformada *Wavelet*, para elevadas taxas de compressão, e quantidade de sinal por ruído similar ao produzido pela transformada *Wavelet*, para níveis de compressão moderados e baixos.

2.3 Considerações

Este capítulo introduziu o dado sísmico que é uma imagem da estrutura geológica da superfície terrestre, destacando sua finalidade e descrevendo os processos de aquisição, processamento e interpretação destes dados. Além disso, foi apresentado o SEG-Y que é um formato de arquivo de dados sísmicos amplamente conhecido e utilizado na troca de dados entre organizações.

Uma das principais características do dado sísmico é o seu tamanho. Como outros tipos de *big data*, o dado sísmico possui grande volume, o que constitui um desafio para seu o armazenamento e transmissão. Todavia, a literatura é vasta no que diz respeito às técnicas de compressão, tanto sem perdas quanto com perdas.

No que diz respeito às técnicas de compressão com perdas, estas proporcionam elevadas taxas de compressão, mas não são amplamente aceitas por geofísicos e geólogos, uma vez que inserem ruído no dado sísmico durante os processos de compressão e descompressão – o que pode comprometer o processamento e interpretação dos dados. Por outro lado, as técnicas de compressão sem perdas proporcionam taxas de compressão pouco expressivas, principalmente quando comparadas a técnicas com perdas, mas são preferenciais quando há a necessidade de reduzir o volume do dado sem descartar parte dele.

Além disso, outro fator que prejudica a adoção de técnicas de compressão com perdas é a escolha de métricas de qualidade para avaliar os impactos da compressão no dado sísmico. Muitos autores utilizam apenas a comparação visual do dado, mas mesmo que o ruído não seja visível, ele pode ser grande e ser amplificado durante o processamento. Outros autores preferem utilizar a taxa de sinal por ruído, mas esta não reflete a correlação dele, característica que deve se considerada sempre.

Por essas razões, esta dissertação trata apenas da compressão sem perdas e, apesar de serem descritas inúmeras técnicas de compressão com perdas, nenhuma delas é utilizada aqui. Ademais, classificando as abordagens de compressão propostas neste trabalho sob o prisma da taxonomia proposta, elas se enquadram no grupo de *separação de sinal, expoente e mantissa*.

Capítulo 3

A Reestruturação de Arquivos SEG-Y

Este capítulo apresenta a proposta de compressão sem perdas de dados sísmicos, representados no formato SEG-Y. Esta proposta inclui três formas de reestruturação de arquivos SEG-Y para reduzir sua entropia e, consequentemente, aumentar as taxas de compressão. Além disso, um compressor específico para arquivos SEG-Y é apresentado, que além de ser *multithread* utiliza os mesmos algoritmos de compressão que o PBZip2. Desta forma, este capítulo lista as tecnologias utilizadas no desenvolvimento do compressor, chamado de SEG-Y-BZ2, e descreve sua arquitetura e interface.

3.1 Proposta

Atualmente, muitos programas de compressão estão disponíveis no mercado e podem ser utilizados na compressão de diversos tipos de dados, como arquivos de texto, áudio e vídeo. Eles apresentam boas taxas de compressão, principalmente se for considerado que foram desenvolvidos para comprimir dados genéricos – qualquer tipo de arquivo.

Boa parte destes programas são gratuitos e estão disponíveis na Internet, o que significa que podem ser adquiridos prontamente sem nenhum custo monetário. Também são facilmente instalados e suas versões contemplam os principais sistemas operacionais existentes. Alguns deles inclusive foram desenvolvidos em versões *multithread*, visando reduzir o tempo de execução dos algoritmos de compressão.

Apesar da facilidade, disponibilidade e baixo custo, pouco se sabe sobre a utilização destes programas na compressão de arquivos SEG-Y. Muitas técnicas de compressão sem e com perdas foram propostas, como mostrado no capítulo anterior, mas não há preocupação quanto ao uso destes compressores. Talvez porque os

arquivos SEG-Y e, consequentemente, os dados sísmicos são compostos em grande parte por amostras – números inteiros ou em ponto flutuante – de grande variação, o que se traduz em baixas taxas de compressão quando nenhum pré-processamento do SEG-Y é realizado.

Todavia, é possível obter melhores taxas de compressão quando o arquivo SEG-Y é reestruturado e tem sua entropia reduzida em relação ao arquivo original. Assim, os compressores presentes no mercado podem produzir boas taxas de compressão e constituem uma boa opção face a complexidade de outras técnicas. Além disso, é possível sobrepor os processos de reestruturação e compressão com intuito de reduzir o tempo de execução dos programas.

Tendo em vista a falta de trabalhos que explorem o uso destes compressores e motivado pela variabilidade, facilidade, disponibilidade e baixo custo deles, propõe-se utilizá-los na compressão de arquivos SEG-Y. Mais do que isso, procura-se desenvolver abordagens de reestruturação do dado sísmico, de maneira a modificar a estrutura dos arquivos SEG-Y e reduzir sua entropia, proporcionando maiores taxas de compressão. Para reduzir o tempo de execução despendido tanto pela reestruturação quanto pela compressão, busca-se sobrepor estes processos através da divisão do arquivo SEG-Y em blocos que podem ser processados em paralelo.

O restante desta seção apresenta o desempenho de diversos compressores presentes no mercado na compressão de arquivos SEG-Y, destacando as taxas de compressão, tempo de compressão e descompressão. Em seguida, serão descritas as propostas para reestruturar o arquivo SEG-Y visando reduzir sua entropia e, finalmente, o desempenho dos programas na compressão dos arquivos SEG-Y reestruturados.

3.1.1 Desempenho de Compressores em Arquivos SEG-Y

Foram selecionados cinco compressores com o objetivo de medir seu desempenho na compressão de arquivos SEG-Y, a saber: 7-Zip[1], BZip2[2], GZip[3], PBZip2[4] e RAR[5]. Todos estão disponíveis para *download* e são gratuitos ou possuem uma versão gratuita para avaliação.

O 7-Zip é um programa gratuito que implementa o algoritmo *Lempel-Ziv-Markov* (LZMA), baseado no LZ77 [59], e atinge boas taxas de compressão. Além do LZMA, as versões mais recentes do programa utilizam outros algoritmos para compressão, como PPMD e *Deflate*. Ele também pode ser utilizado na compressão de um ou mais arquivos e diretórios inteiros.

[1] `http://www.7-zip.org/`
[2] `http://www.bzip.org/`
[3] `http://www.gzip.org/`
[4] `http://compression.ca/pbzip2/`
[5] `http://www.rarlab.com/`

O BZip2 também é um programa gratuito, mas que utiliza a transformada *Burrows-Wheeler* [60] e *Huffman* na compressão. Diferentemente do 7-Zip, este programa opera em um único arquivo por vez, não sendo possível a compressão de diretórios.

O GZip, ou, GNU Zip é um programa incorporado aos sistemas Unix, livre de patentes, implementa o algoritmo *Deflate*, que também é uma variação do LZ77, e *Huffman*. Ele também permite a compressão de arquivos e diretórios.

O PBZip2 é uma versão *multithread* do BZip2. Ele emprega os mesmos algoritmos na compressão, mas divide um arquivo em blocos de tamanho igual que são comprimidos em paralelo e independente uns dos outros.

Por fim, o RAR é um programa comercializado sob licença não gratuita, mas que disponibiliza uma versão de avaliação. Justamente por produzir arquivos em formato proprietário, não se sabe ao certo quais algoritmos de compressão ele usa. Porém, assim como outros programas, o RAR implementa a compressão em uma ou mais *threads*. Nos experimentos realizados aqui, RAR utiliza apenas uma *thread*.

Além desses compressores apresentados anteriormente, os compressores CMIX[6], PAQ8[7] e FPAQ8_v3[8] foram experimentados na compressão de dados sísmicos. O CMIX é um compressor que requer elevado uso de processamento CPU e memória, mas é otimizado para produzir as melhores taxas de compressão. Além disso, ele possui os melhores resultados na compressão de texto, sendo apontado como o estado da arte na compressão em diversos *sites*. Todavia, não foi possível avaliar o desempenho deste compressor em dados sísmicos, pois, após algumas horas de processamento e progresso 0%, o sistema operacional eliminou o processo do CMIX por falta de memória.

O PAQ8 é um compressor que experimenta diversos algoritmos de compressão existente para alcançar elevadas taxas de compressão em detrimento do tempo de execução e uso de memória. Ele foi utilizado na compressão de um arquivo SEG-Y com tamanho de 2 Gigabytes. Diferentemente do CMIX, o PAQ8 requer menos processamento e memória, porém, depois de mais de 30 horas de execução e sem indícios de que o programa estava próximo de finalizar a compressão, decidiu-se interromper sua execução. Desta forma, conclui-se que este compactador é inviável na compressão de SEG-Y devido ao tempo de execução.

Já o FPAQ8_v3, ou *Fast PAQ8*, é uma versão de menor tempo de execução do PAQ8. Esta versão permite escolher o nível de compressão que vai da execução mais rápida e piores taxas de compressão até a execução mais lenta e melhores taxas de compressão. Nos experimentos realizados, um arquivo SEG-Y de 2 Gigabytes foi

[6] `http://www.byronknoll.com/cmix.html`
[7] `http://mattmahoney.net/`
[8] `http://mattmahoney.net/dc/text.html`

comprimido com a opção de maior velocidade de compressão. Mesmo assim, depois de mais de 5 horas de processamento, a compressão ainda não havia sido finalizada.

Portanto, o conjunto de compressores utilizados nos experimentos foi composto apenas por: 7-Zip, BZip2, GZip, PBZip2 e RAR. Estes, por sua vez, não apresentam as melhores taxas de compressão entre os compressores existentes, mas possuem tempo de execução viável para a compressão de dados sísmicos.

Os experimentos foram executados em um computador com sistema operacional Ubuntu 14.04 LTS, processador Intel Core i7, 8 Gigabytes de memória RAM, disco rígido de 500 Gigabytes e 7200 rpm. Além disso, foram utilizados 14 arquivos SEG-Y sintéticos nestes experimentos, ou seja, gerados artificialmente e que são disponibilizados pelo site da SEG[9]. Foram utilizados os arquivos de "2004 BP Velocity Analysis Benchmark"[10] e "Statics 94"[11]. Todos são dados sísmicos de duas dimensões e tamanho médio de 1.83 Gigabytes. A escolha de SEG-Y sintéticos e não os reais nesta fase de experimentos foi tomada, pois os arquivos sintéticos, apesar de não possuírem tanto ruído quanto os reais, são bem menores e permitem que os experimentos sejam realizados mais rapidamente.

Das métricas utilizadas para avaliar o desempenho dos compressores listados anteriormente, foram computados o percentual que o tamanho do arquivo comprimido representa do SEG-Y sem compressão – tamanho do arquivo comprimido sobre o tamanho do SEG-Y sem compressão –, o tempo de compressão e o tempo de descompressão. Além disso, como o PBZip é *multithread*, seus tempos de compressão e descompressão serão comparados apenas com os tempos do BZip2 que é sua versão sequencial.

A Figura 3.1 exibe o desempenho dos programas quanto ao tamanho do arquivo comprimido. Como pode ser observado, boa parte dos compressores tem desempenho similar, produzindo arquivos comprimidos com aproximadamente 70% do tamanho do SEG-Y sem compressão, ou seja, 1.28 Gigabytes. Porém, o compressor 7-Zip obtém o melhor desempenho, sendo capaz de reduzir o tamanho do SEG-Y a 65.43% do seu tamanho sem compressão, o que equivale a um arquivo de 1.2 Gigabytes.

Também se nota que os tamanhos dos arquivos comprimidos por BZip2 e PBZip2 não são idênticos, devido ao paralelismo do PBZip2. Como era de se esperar, ao priorizar a redução do tempo de compressão, dividindo o arquivo em blocos que são comprimidos em paralelo, o compressor PBZip2 aumenta o tamanho do arquivo comprimido em relação ao BZip2. Apesar deste aumento não ser significativo, ele pode crescer a medida em que o número de blocos cresce.

[9] `http://software.seg.org/datasets/2D/index.html`
[10] `http://software.seg.org/datasets/2D/2004_BP_Vel_Benchmark/`
[11] `http://software.seg.org/datasets/2D/Statics_1994/`

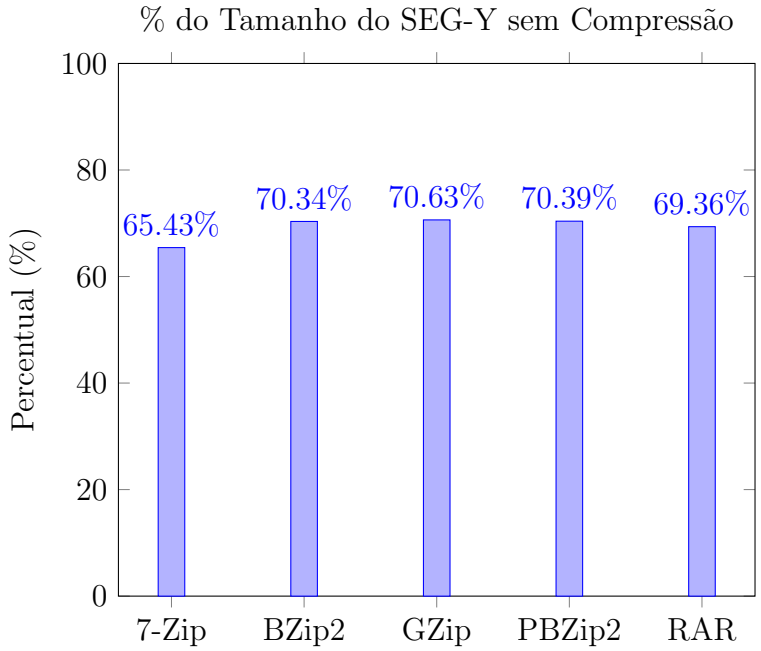


Figura 3.1: Tamanho de SEG-Y sintéticos comprimidos em relação aos arquivos sem compressão.

Quanto aos tempos de compressão e descompressão, observa-se que o GZip é o compressor mais rápido, dentre as versões sequenciais – 7-Zip, BZip2, GZip e RAR. Por outro lado, apesar do 7-Zip produzir o menor arquivo, ele é o compressor que demanda mais tempo durante a compressão, exigindo mais de 4 min, seguido por RAR (3 minutos e 24 segundos) e BZip2 (2 minutos e 58 segundos) – ver Figura 3.2.

Comparando o BZip2 e PBZip2, ou seja, a versão sequencial e paralela (com oito *threads*) de um mesmo algoritmo de compressão, a diferença no desempenho é notável no que diz respeito aos tempos de compressão e descompressão – ver Figura 3.3. Apesar dos programas produzirem arquivos comprimidos de tamanho similar – a diferença é muito pequena neste quesito –, a redução nos tempos de compressão e descompressão é significativa. Assim, o PBZip2 é capaz de comprimir e descomprimir um arquivo SEG-Y cerca de 78% mais rápido do que o BZip2 e 55% mais rápido do que o GZip, o que o torna o compressor mais rápido de todos experimentados.

Como foi observado nestes primeiros experimentos, os programas avaliados possuem desempenho variado na compressão de arquivos SEG-Y. A maior parte deles apresenta taxas de compressão pouco expressivas e são lentos em demasiado, demandando mais de 1 minuto na compressão de arquivos com quase 2 Gigabytes de tamanho. Arquivos muito pequenos, tendo em vista que podem atingir Terabytes.

Por outro lado, o PBZip2 se mostrou uma boa opção na compressão. Ele é capaz de reduzir o tempo de execução, devido ao paralelismo, sem afetar significativamente as taxas de compressão. Todavia, seria bom reduzir o tamanho do arquivo compri-

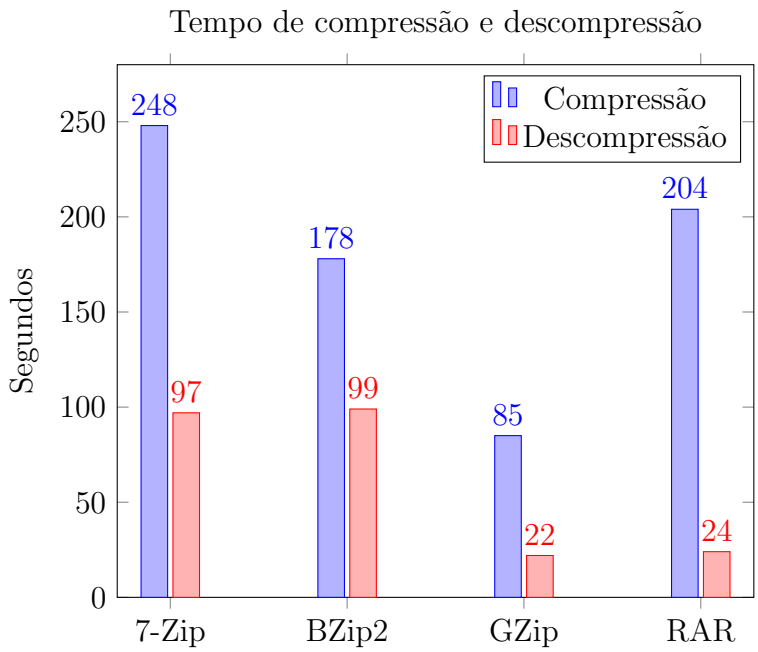


Figura 3.2: O gráfico mostra o tempo de compressão e descompressão, medido em segundos, dos diferentes compressores quando aplicados sobre SEG-Y sintéticos.

mido de todos os compressores, o que pode ser alcançado através da reestruturação do SEG-Y, como é mostrado nas próximas seções.

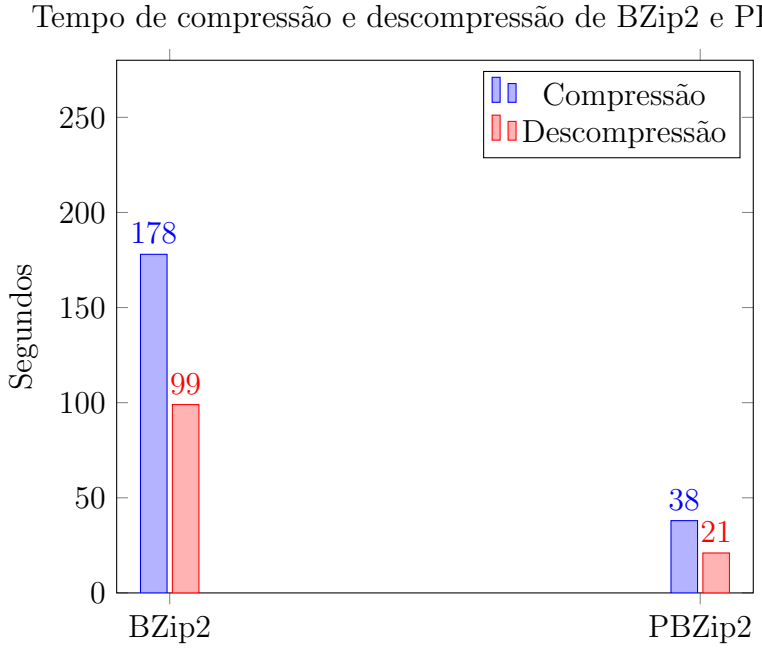


Figura 3.3: O gráfico mostra o tempo de compressão e descompressão, medido em segundos, de BZip2 (sequencial) e PBZip2 (*multithread*) quando aplicados sobre SEG-Y sintéticos.

3.1.2 Propostas para Reestruturação de Arquivos SEG-Y

Como foi observado na seção anterior, é possível utilizar diversos programas gratuitos para comprimir arquivos SEG-Y, desde que estes sejam eficientes tanto em tempo de execução quanto em taxa de compressão. No que diz respeito ao tempo, que é uma restrição nas aplicações sísmicas, o PBZip2 se mostrou a melhor opção, visto que é um compressor *multithread*. Resta então melhorar as taxas de compressão, o que pode ser alcançado através da reestruturação do arquivo SEG-Y.

A reestruturação do arquivo SEG-Y é um pré-processamento que deve ser realizado antes da compressão, cujo objetivo é reduzir a entropia do arquivo SEG-Y para aumentar as taxas de compressão. Como o arquivo SEG-Y é composto por dois tipos de dados – cabeçalhos e amostras, ver seção 2.1 –, propõe-se que ambos sejam comprimidos em separado. Porém, os cabeçalhos devem ser comprimidos sem nenhum pré-processamento adicional, enquanto que as amostras devem ser reestruturadas.

A decisão de não realizar um pré-processamento específico para os cabeçalhos foi tomada por duas razões. Primeiro porque os programas são eficientes na compressão desta parte do arquivo SEG-Y e segundo porque os cabeçalhos representam uma pequena fração do tamanho do arquivo. No caso dos arquivos utilizados nos experimentos da seção 3.1.1, os cabeçalhos representam apenas 5% do seu tamanho, ou seja, 94 Megabytes – considerando que o tamanho médio do SEG-Y é de 1.83 Gigabytes.

Por outro lado, as amostras que representam grande parte do tamanho do SEG-Y – 95% do seu tamanho para os arquivos utilizados na seção 3.1.1, ou seja, 1.74 Gigabytes aproximadamente – possuem grande variação que impactam nas taxas de compressão. Por essa razão, elas precisam ser submetidas a um pré-processamento que reduza sua entropia.

Os resultados da compressão dos cabeçalhos e amostras dos arquivos SEG-Y utilizados na seção 3.1.1 é ilustrado na Figura 3.4. O arquivo de amostras comprimidas fica com 71-73% do tamanho do arquivo de amostras sem compressão, enquanto que o arquivo de cabeçalhos comprimidos tem 4-5% do tamanho do arquivo de cabeçalhos sem compressão. Desta forma, corrobora-se a necessidade de reestruturar as amostras com o intuito de reduzir sua entropia, enquanto que os cabeçalhos já obtêm elevadas taxas de compressão.

Para a reestruturação da amostras, considera-se que estas são números em ponto flutuante de 32 bits cuja arquitetura é IBM – ver Figura 2.2. Nesta arquitetura, o primeiro bit corresponde ao sinal, os sete bits seguintes ao expoente e os 24 bits restantes à mantissa. As abordagens que são apresentadas aqui dividem a mantissa em duas partes: a parte alta, que corresponde aos bits mais significativos (da mantissa), e a parte baixa que corresponde aos bits restantes. Desta forma, propõe-se

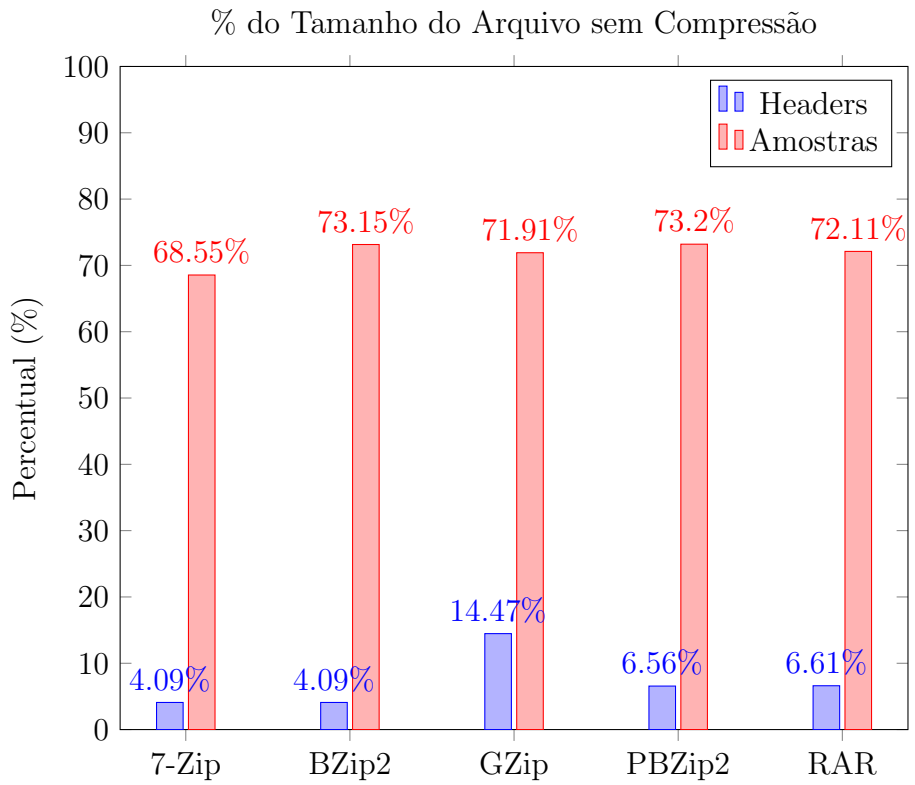


Figura 3.4: O gráfico mostra o percentual que o tamanho do arquivo comprimido representa do tamanho do arquivo sem compressão.

três esquemas de reestruturação de *float* (as amostras sísmicas) que separam seus bits em três partes e as comprimem separadamente:

1. Expoente (7 bits), mantissa alta que inclui o bit de sinal (9 bits) e mantissa baixa (16 bits);

2. Expoente que inclui o bit de sinal (8 bits), mantissa alta (8 bits) e mantissa baixa (16 bits);

3. Expoente que inclui o bit de sinal (8 bits), mantissa alta (16 bits) e mantissa baixa (8 bits).

A Figura 3.5 ilustra as três abordagens de reestruturação, delimitando os bits que compõem cada uma das três partes em que se dividem uma amostra. Por exemplo, na primeira reestruturação, as amostras de 32 bits e formato IBM, são divididas em três partes: parte 1 composta pelo expoente (bits 1 a 7); parte 2 composta por sinal e mantissa alta (bit 0 e bits 8 a 15); e parte 3 que é composta pela mantissa baixa (bits 16 a 31). Todas as amostras são divididas nestas três partes que, por sua vez, são armazenadas em vetores, sendo estes comprimidos separadamente.

Na primeira reestruturação, junta-se o bit de sinal com os oito bits da mantissa alta, pois como o sinal é da mantissa e não do expoente, acredita-se que exista uma

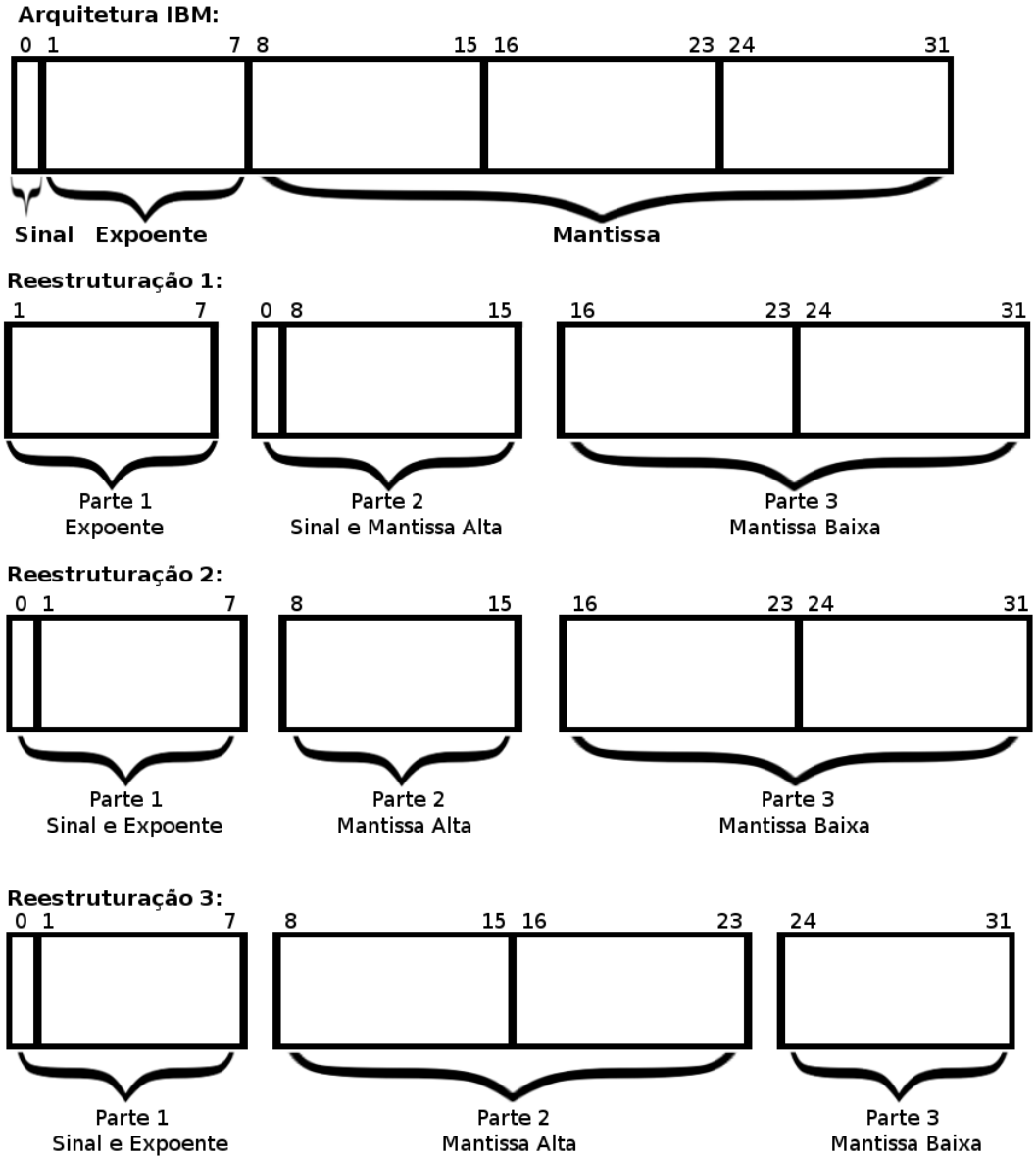


Figura 3.5: A figura ilustra o formato IBM de números em ponto flutuante de 32 bits e as estratégias propostas para reestruturação de amostras.

Parte	Tamanho (bytes)	bits	Variância
Amostra Inteira	4	0-31	25292.59
Sinal e Expoente	1	0-7	6993.93
Mantissa Alta	1	8-15	7050.29
Mantissa Baixa	1	16-23	6731.92
Mantissa Baixa	1	24-31	5658.89

Tabela 3.1: Comparação entre a variância das amostras e de suas partes – expoente, mantissa alta e mantissa baixa.

correlação entre estes valores. Além disso, tanto a primeira quanto a segunda reestruturação possuem 16 bits de mantissa baixa, pois como o dado sísmico é formado por ruído aleatório, os bits menos significativos da mantissa são basicamente formados por ruído e, consequentemente, alcançam pequena compressão. Desta forma, separar a parte alta da parte baixa da mantissa permite reduzir a entropia do arquivo SEG-Y.

Na terceira reestruturação, a mantissa alta é formada por 16 bits e a mantissa baixa por apenas oito. Neste caso, acredita-se que os instrumentos utilizados na aquisição sísmica possuem menos de 32 bits de precisão. Desta forma, os bits menos significativos da mantissa são formados por zeros e, consequentemente, alcançam boas taxas de compressão quando separados dos demais.

Ainda utilizando os arquivos SEG-Y sintéticos da seção 3.1.1, computou-se a variação das amostras como um todo e das suas partes: sinal e expoente (1 byte), mantissa alta (1 byte) e mantissa baixa (2 bytes). No caso da mantissa baixa, a variação foi computada em separado para cada byte. O resultado é apresentado na Tabela 3.1.

Como pode ser observado, a amostra em sua representação original (4 bytes) possui maior variância que suas partes independentes. Desta forma, a separação é benéfica para a compressão e permite alcançar maiores taxas de compressão. Todavia, é preciso lembrar que como os dados são sintéticos, é possível que eles possuam menos ruído do que dados sísmicos reais, ou até mesmo não possuam ruído algum em sua composição.

3.1.3 Desempenho de Compressores após Reestruturação de Arquivos SEG-Y

Para comprovar que a reestruturação é capaz de reduzir o tamanho de arquivos comprimidos, a segunda proposta de reestruturação de amostras apresentada na seção anterior – que separa expoente (1 byte, incluindo o bit de sinal), mantissa alta (1 byte) e mantissa baixa (2 bytes) – foi aplicada aos arquivos SEG-Y utilizados na seção 3.1.1. Em seguida, os arquivos reestruturados foram submetidos aos programas

apresentados anteriormente.

Analisando os resultados ilustrados na Figura 3.6, observa-se que a reestruturação do arquivo aumentou de fato a eficiência dos programas em taxa de compressão. Desta forma, é possível comprimir o SEG-Y e atingir cerca de 60% do seu tamanho original, sem causar danos no dado. Esta redução é considerável, principalmente se o tamanho de arquivos SEG-Y reais for levada em conta.

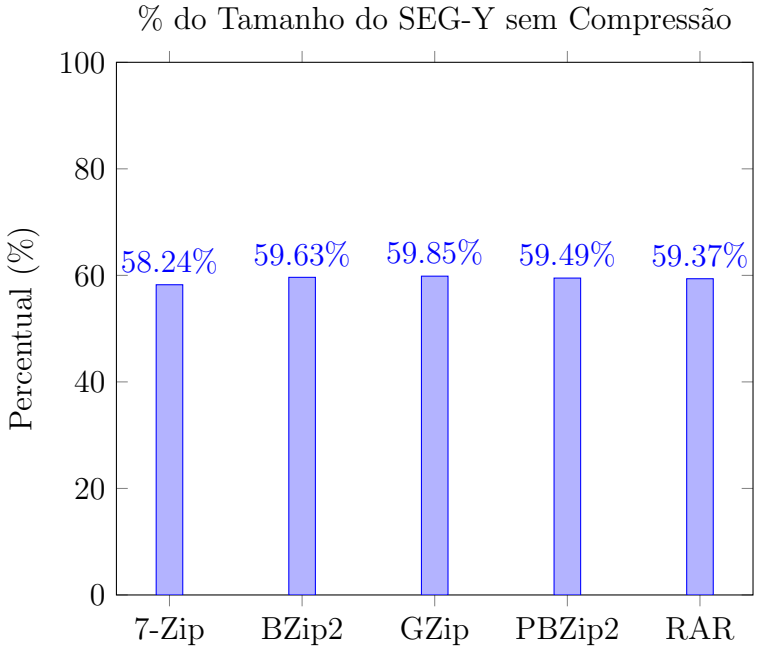


Figura 3.6: O gráfico mostra o percentual que o tamanho do arquivo reestruturado e comprimido representa do tamanho do SEG-Y sem compressão.

O maior problema desta abordagem diz respeito ao tempo de execução, uma vez que a reestruturação é realizada completamente independente da compressão/descompressão. Desta forma, o tempo de execução do conjunto (reestruturação e compressão/descompressão) pode aumentar significativamente. Nos experimentos realizados, por exemplo, o tempo médio para reestruturação dos arquivos SEG-Y sintéticos é de cerca de 1 minuto – tempo que aumenta à medida que o tamanho do arquivo SEG-Y cresce.

Portanto, é preciso sobrepor os processos de reestruturação e compressão/descompressão para obter (1) melhores taxas de compressão e (2) diminuir o tempo de execução do conjunto. A estratégia utilizada para atingir estes objetivos é baseada no PBZip2 que procura dividir o arquivo SEG-Y em vários blocos que são reestruturados e comprimidos/descomprimidos paralelamente. Para provar que esta estratégia é eficiente, o compressor SEG-Y-BZ2 foi desenvolvido e é apresentado na seção seguinte. Utilizando a reestruturação e compressão de blocos de arquivos SEG-Y em paralelo, o modelo proposto supera o desempenho do PBZip2, tanto em tempo quanto em taxa de compressão – é possível reduzir em até 20% o tamanho

de arquivos SEG-Y comprimidos.

3.2 O Compressor SEG-Y-BZ2

Um compressor de SEG-Y foi desenvolvido neste trabalho e chamado de SEG-Y-BZ2. Ele é *multithread*, implementa a reestruturação do arquivo e a compressão utilizando os mesmos algoritmos que o BZip2 e PBZip2.

O paralelismo garante que a compressão e a descompressão serão eficientes, diminuindo consideravelmente o tempo de execução sem afetar significativamente as taxas de compressão. Para tal, o arquivo SEG-Y é dividido em blocos de tamanho fixo que são processados simultaneamente e independentes uns dos outros.

Já a reestruturação, como mostrado anteriormente, pode diminuir a entropia do arquivo SEG-Y e aumentar a eficiência do processo de compressão. Desta forma, arquivos SEG-Y reestruturados e comprimidos podem ter tamanho menor do que SEG-Y que são comprimidos em sua estrutura original. Além disso, o paralelismo permite que diversos blocos do arquivo SEG-Y sejam reestruturados e comprimidos simultaneamente, reduzindo o tempo de execução da aplicação.

Finalmente, a compressão com os algoritmos utilizados no BZip2 e PBZip2 foi escolhida, pois a taxa de compressão encontrada com estes compressores é semelhante a taxa de compressão dos demais. Além disso, como o PBZip2 é o compressor mais eficiente em tempo de compressão e descompressão dentre os selecionados, deseja-se comparar seu desempenho com o desempenho alcançado pelo SEG-Y-BZ2. Todavia, qualquer algoritmo de compressão pode ser implementado.

As seções seguintes descrevem a arquitetura do SEG-Y-BZ2, as tecnologias utilizadas na sua implementação e a interface do programa.

3.2.1 Arquitetura

O compressor SEG-Y-BZ2 foi desenvolvido com o objetivo exclusivo de comprimir arquivos SEG-Y. Como pode ser visto na Figura 3.7, um arquivo SEG-Y que contém cabeçalhos, traços sísmicos e amostras é submetido ao compressor SEG-Y-BZ2 que, internamente, reestrutura o SEG-Y e o comprime com os mesmos algoritmos utilizados pelo BZip2 e PBZip2. Ao final da execução, o programa gera um arquivo comprimido de tamanho inferior ao tamanho do SEG-Y sem compressão.

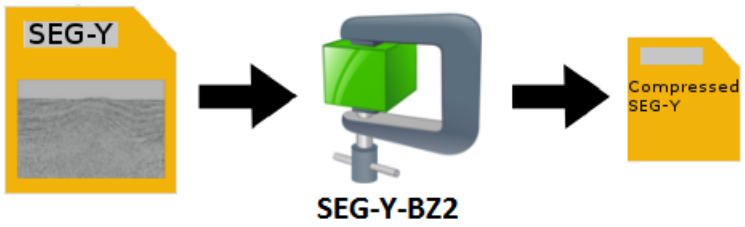


Figura 3.7: Esquema de compressão de SEG-Y no SEG-Y-BZ2.

O funcionamento do compressor é muito simples e é ilustrado na Figura 3.8. No início do processo de compressão, os *buffers*, filas e outras estruturas utilizadas pela aplicação são inicializadas. Em seguida, um bloco de dados que armazena os 3600 bytes iniciais de cabeçalhos (*headers*) do arquivo SEG-Y é criado e adicionado à fila de compressão. Também são criadas três "tipos" de *threads*: (1) uma *thread* para leitura e reestruturação dos traços do arquivo SEG-Y, (2) uma *thread* para escrita do arquivo comprimido e (3) N *threads* de compressão – o número de *threads* de compressão depende da quantidade de núcleos de processamento disponíveis.

A *thread* de leitura e reestruturação é responsável por ler uma quantidade pré-determinada de traços do arquivo, separar os cabeçalhos dos traços e amostras, e reestruturar as amostras. Assim, são gerados quatro blocos de dados a serem comprimidos, sendo um bloco composto por cabeçalhos dos traços e três blocos cada um com parte dos bits das amostras. Por sua vez, as *threads* de compressão são responsáveis por comprimir os blocos de dados gerados pela *thread* de leitura que, por fim, são escritos no arquivo comprimido pela *thread* de escrita.

O fluxo de compressão é descrito a seguir:

1. Inicialização dos *buffers*, filas e demais estruturas;

2. Criação de um bloco que contém os 3600 bytes iniciais de cabeçalho textual e cabeçalho binário do arquivo SEG-Y e inclusão do bloco na fila de compressão;

3. Criação das *threads* de leitura e reestruturação de traços do SEG-Y, compressão e escrita do arquivo comprimido;

4. A *thread* de leitura lê uma quantidade pré-determinada de traços do arquivo SEG-Y, reestrutura-os, cria tantos blocos quanto necessário – um bloco só de cabeçalhos, um bloco só com expoentes, um bloco só com mantissa alta e um bloco só com mantissa baixa, por exemplo – e adiciona-os na fila de compressão;

5. Cada *thread* de compressão que está ociosa, retira um bloco da fila de compressão, comprime-o e inclui o bloco comprimido na fila de escrita;

6. A *thread* de escrita retira os blocos da fila de escrita e escreve-os no arquivo comprimido;

7. Ao final do processo, os *buffers*, filas e demais estruturas são desalocados.

Portanto, a *thread* de leitura, que lê o arquivo SEG-Y, reestrutura-o e cria blocos a serem comprimidos, alimenta a fila da compressão que é do tipo FIFO – *First In First Out*. As *threads* de compressão consomem esta fila e alimentam a fila da escrita, que também é do tipo FIFO. Por fim, a *thread* de escrita consome esta última fila e escreve os blocos comprimidos no arquivo de saída. Estas ações de criação, compressão e escrita de blocos, que correspondem aos passos 4, 5 e 6 do fluxo anterior, são executados simultaneamente e em paralelo.

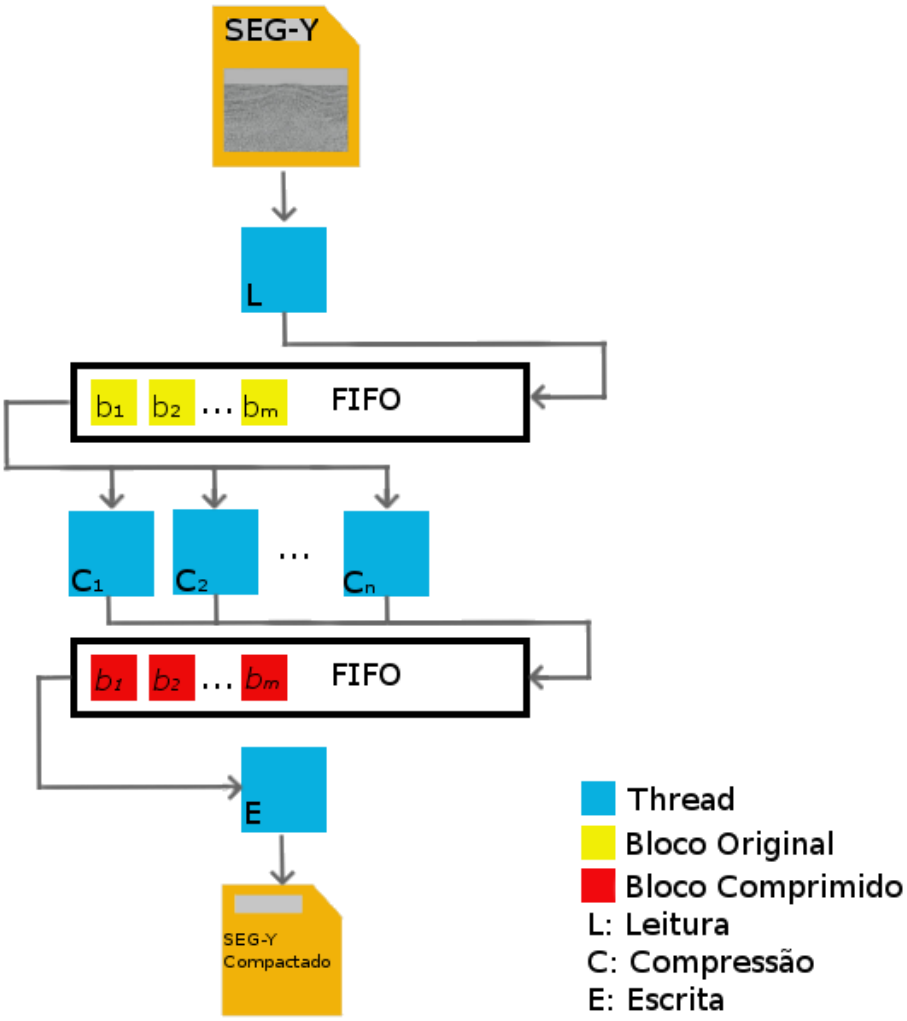


Figura 3.8: Arquitetura do compressor SEG-Y-BZ2 durante o processo de compressão.

Comparativamente falando, o fluxo de compressão do SEG-Y-BZ2 e do PBZip2 são idênticos, pois possuem *threads* de leitura, escrita e compressão. A diferença é que a *thread* de leitura do PBZip2 apenas lê blocos do arquivo SEG-Y e os insere

na fila de compressão, enquanto que a *thread* de leitura do SEG-Y-BZ2 lê os blocos do arquivo SEG-Y, reestrutura-os e insere-os na fila de compressão.

Com o paralelismo, à medida que os blocos são incluídos nas filas, eles vão sendo consumidos pelas respectivas *threads*, tornando possível sobrepor os processos de reestruturação, compressão e escrita, e reduzir o tempo de execução. Desta forma, melhora-se o desempenho do SEG-Y-BZ2.

Quanto ao processo de descompressão, este possui estrutura similar à compressão. A diferença é que existe uma nova *thread* para restaurar a estrutura original do SEG-Y e uma nova fila FIFO de restauração que contém blocos descomprimidos, que devem ser combinados para formar a estrutura original do SEG-Y – ver Figura 3.9.

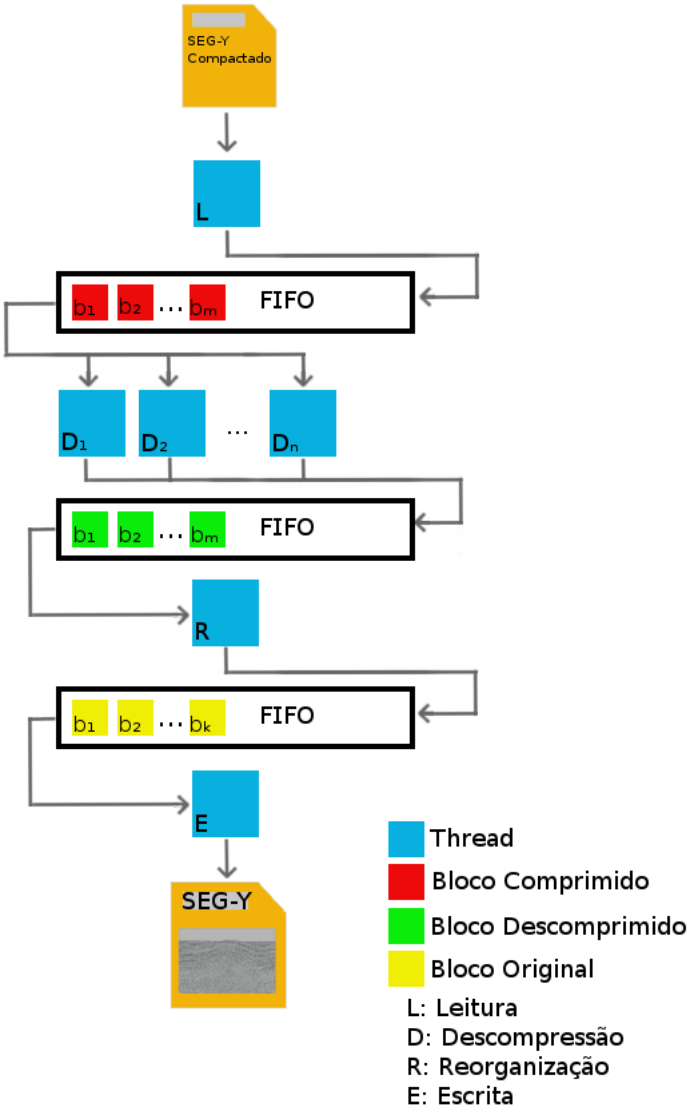


Figura 3.9: Arquitetura do compressor SEG-Y-BZ2 durante o processo de descompressão.

Portanto, durante a descompressão, a *thread* de leitura lê os blocos comprimidos

e os inclui na fila de descompressão. Por sua vez, as *threads* de descompressão consomem a esta fila, descomprimem os blocos e os inserem na fila de restauração. A *thread* de restauração consome os blocos desta fila, restaura a estrutura original do bloco do SEG-Y e inclui o bloco restaurado na fila de escrita. Por fim, a *thread* de escrita consome os blocos desta última fila e escreve-os no arquivo SEG-Y. Ao final do processamento, o SEG-Y é descomprimido e tem sua estrutura original restaurada.

A *thread* de restauração consome um número fixo de blocos por vez que depende da abordagem de reestruturação aplicada. Como todas as abordagens separam as amostras em três partes, a *thread* de restauração consome quatro blocos por vez, sendo um bloco de cabeçalhos e três que contém bits das amostras, e produz um único bloco para ser escrito.

A maior diferença entre os fluxos de descompressão do SEG-Y-BZ2 e PBZIP2 é a existência da *thread* e fila de restauração da estrutura original do SEG-Y. Neste caso, o SEG-Y-BZ2 precisa destas estruturas para restabelecer o formato original do arquivo SEG-Y, enquanto que o PBZip2 que não realiza a reestruturação, dispensa estes elementos no seu fluxo de descompressão, sendo este fluxo idêntico ao fluxo de compressão.

3.2.2 Tecnologias

O compressor SEG-Y-BZ2 foi desenvolvido utilizando-se as seguintes tecnologias:

1. Linguagem C++;
2. Biblioteca *pthread*;
3. Biblioteca semaphore; e
4. Biblioteca libbzip2[12].

Destas, a biblioteca libbzip2 implementa os métodos de compressão e descompressão utilizados por BZip2 e PBZip2. Existem três níveis de funções de compressão: baixo, alto e utilidade. O nível baixo disponibiliza funções para alocação e desalocação das estruturas de compressão e descompressão. Já o nível alto permite a manipulação de arquivos no formato .bz2, gerados pelos compressores BZip2 e PBZip2. Por fim, o nível utilidade, que foi usado no compressor SEG-Y-BZ2, permite a compressão e descompressão de *buffers* em memória cuja alocação e desalocação é de responsabilidade do programa que utiliza a biblioteca.

[12]http://www.bzip.org/1.0.5/bzip2-manual-1.0.5.html

3.2.3 Interface

O compressor SEG-Y-BZ2 possui interface em linha de comando e pode ser chamado em um terminal UNIX (ou Windows) da seguinte forma:

$./SEG-Y-BZ2 --help

-c Para indicar a compressão do arquivo de entrada. Este arquivo deve ser um SEG-Y com extesão .sgy ou .segy.

-d Para indicar a descompressão do aquivo de entrada. Este arquivo deve ser um SEG-Y comprimido com extensão .rbz2.

-i DIR/NOME Para indicar o diretório do arquivo de entrada, concatenado com o nome do arquivo.

-o DIR/NOME Para indicar o diretório do arquivo de saída, concatenado com o nome do arquivo. Caso nenhum nome seja informado, o programa salva o arquivo de saída com o nome do arquivo de entrada no diretório especificado por este comando.

-l DIR/NOME Para especificar o diretório, concatenado com o nome do arquivo em que se deseja salvar os logs impressos com o parâmetro -v.

-t NÚMERO Para especificar o número máximo de traços por bloco. Por padrão, ó número máximo de traços é 10000.

-b NÚMERO Para especificar o número máximo de blocos que podem ser alocados em memória. Por padrão, são alocados no máximo 20 blocos.

-n NÚMERO Para especificar o número de *threads* de compressão ou descompressão que devem ser criadas. Caso este parâmetro não seja utilizado, o número de *threads* criado é igual ao número de *cores* de processamento disponíveis.

-v Para verbosidade.

-p Para ocultar o progresso da compressão ou descompressão.

Um exemplo de comando que realiza a compressão do arquivo SEG-Y de nome *mig.sgy*, salva o arquivo comprimido com nome *mig.rbz2*, utilizando 5000 traços por bloco, 10 blocos em memória e habilitando *verbosity*, pode ser visto a seguir:

$./SEG-Y-BZ2 -v -c -i ./mig.sgy -o ./mig.sgy.rbz2 -t 5000 -b 10

Na descompressão, não é possível utilizar o parâmetro -t que especifica o número de traços, pois o programa obtém esta informação do arquivo comprimido. Portanto, um comando que descomprime um arquivo, utilizando no máximo 15 blocos em memória e habilitando *verbosity*, pode ser visto a seguir:

$./SEG-Y-BZ2 -v -d -i ./mig.rbz2 -o ./mig.sgy -b 15

Quanto ao consumo máximo de memória, este depende de três fatores, dos quais dois são parâmetros do programa: o número máximo de blocos em memória e o número máximo de traços lidos. Já o terceiro fator depende do arquivo SEG-Y e é o número de amostras por traço sísmico. Assim, a quantidade máxima de memória utilizada é calculada por: $M_{max} = 4 * A * T * B$. Em que M_{max} representa a quantidade máxima de memória em bytes, A representa o número de amostras em cada traço do SEG-Y, T o número de traços por bloco e B o número máximo de blocos em memória. O valor 4 na equação equivale ao número de bytes das amostras, que são do tipo *floats*.

Por exemplo, para um arquivo SEG-Y que possui 1500 amostras em cada traço, que é comprimidodo com parâmetros *-t 5000* e *-b 10*, a quantidade máxima de memória é 286.10 Megabytes.

Capítulo 4

Experimentos e Resultados

Este capítulo apresenta os experimentos realizados com o compressor SEG-Y-BZ2, descrevendo o ambiente de execução, dados utilizados, métricas de avaliação e otimização de parâmetros. O desempenho do compressor é então avaliado quanto às formas de reestruturação propostas, mostrando que a melhor abordagem de reestruturação consiste em separar as amostras em expoente (incluindo bit de sinal), mantissa alta e mantissa baixa, e comprimir estas partes separadamente. Além disso, esta abordagem supera o desempenho de outras abordagens que foram propostas na literatura e o compressor SEG-Y-BZ2 é capaz de reduzir o tamanho do arquivo comprimidoado em até 20%, comparando com os resultados alcançados com PBZip2, além de apresentar menor tempo de execução.

4.1 Dados Sísmicos

Os dados sísmicos utilizados nos experimentos foram cedidos pelo Banco de Dados de Exploração e Produção (BDEP) / Agência Nacional de Petróleo, Gás Natural e Biocombustíveis (ANP). Trata-se de sete arquivos SEG-Y que representam a Bacia do Paraná, obtidos em 2010 via aquisição 2D e são do tipo *pre-stack*.

Juntos, eles totalizam 311.37 Gigabytes de dados e possuem tamanho médio de 44.48 Gigabytes – o maior arquivo tem 53.08 Gigabytes e o menor arquivo possui 35.62 Gigabytes. Além disso, alguns arquivos tem mais de 5 milhões de traços sísmicos com exatamente 2501 amostras em cada um, estruturadas no formato IBM (representado na Figura 2.2).

4.2 Ambiente de Execução

O ambiente de execução dos testes é formado por um computador com a seguinte configuração:

1. Processador Intel Core i7-2600 de 3.40 Ghz, com 4 núcleos de processamento e Hyper-Threading;
2. 8 Gigabytes de memória RAM;
3. Placa de vídeo Geforce GTX-480;
4. HD SATA 2 de 500 Gigabytes e 7200 RPM; e
5. Sistema Operacional Ubuntu 14.04 LTS.

4.3 Compressores e Abordagens de Reestruturação

Apenas dois compressores foram utilizados nos experimentos: o compressor SEG-Y-BZ2, proposto neste trabalho, e o compressor PBZip2. Ambos utilizam os mesmos algoritmos de compressão, que inclui a transformada *Burrows-Wheeler* e *Huffman*, tornando possível comparar seus desempenhos. Porém, o SEG-Y-BZ2 implementa a reestruturação de arquivos SEG-Y, enquanto o PBZip2 não o faz.

Além disso, o compressor SEG-Y-BZ2 é apresentado em várias versões, cada uma implementando uma abordagem de reestruturação de arquivo SEG-Y diferente, sejam estas propostas neste trabalho ou propostas na literatura. Conforme apresentado na seção 3.1.2 e ilustrados na Figura 3.5, as reestruturações propostas neste trabalho são:

1. Separar amostras em expoente (7 bits), mantissa alta que inclui o bit de sinal (9 bits) e mantissa baixa (16 bits);
2. Separar amostras em expoente que inclui o bit de sinal (8 bits), mantissa alta (8 bits) e mantissa baixa (16 bits);
3. Separar as amostras em expoente que inclui o bit de sinal (8 bits), mantissa alta (16 bits) e mantissa baixa (8 bits).

Quanto às abordagens de reestruturação de amostras encontradas na literatura, aquela utilizada por Xie e Qin em [27] foi empregada nos experimentos para fins comparativos. Nela, os autores separam uma amostra em sinal (1 bit), expoente (7 bits) e mantissa (24 bits) e utilizam estas informações num esquema de compressão que envolve o uso da diferença de amostras e o emprego do algoritmo *Arithmetic Coding* – para mais informações, consulte a seção 2.2.1.1. Contudo, esta abordagem de reestruturação foi adaptada para juntar o bit de sinal e os bits do expoente nos experimentos realizados. Assim, o SEG-Y-BZ2 que implementa esta reestruturação

de amostras, separam-nas em expoente que inclui o bit de sinal (8 bits) e mantissa (24 bits), comprimindo estas partes separadamente.

Outra forma de representação de amostras propostas na literatura é através do resíduo da diferença de amostras consecutivas (num mesmo traço sísmico) [21, 22]. Ou seja, dada uma amostra, esta abordagem propõe representá-la pelo resultado da diferença entre seu valor e o valor de amostras anteriores. Nestes experimentos, a fórmula para cálculo da diferença das amostras consiste em subtrair duas amostras consecutivas. Assim, a primeira amostra do traço é representada pelo seu valor, sem alteração. Já as demais amostras são representadas pela subtração entre seu valor e o valor da amostra antecessora.

O maior problema em se utilizar uma fórmula de diferença que envolva várias amostras e coeficientes, como a usada por Xie e Qin, é que ela está estritamente relacionada à curva formada pelo traço sísmico. Assim, uma fórmula que foi construída baseada num arquivo SEG-Y, pode produzir resultados ruins em outros arquivos SEG-Y. Portanto, para utilizar fórmulas mais complexas, é preciso ter conhecimento prévio do dado sísmico, ou utilizar uma fórmula dinâmica que se adapte ao dado em questão – o que aumenta o tempo de execução do compressor.

Por outro lado, a diferença entre amostras consecutivas é relativamente simples e pode produzir bons resultados, desde que as amostras num mesmo traço não variem demasiadamente. Além disso, como a diferença apresenta desempenho semelhante a predição linear e é mais rápida de ser computada, este esquema de reestruturação foi utilizado em uma das versões do SEG-Y-BZ2 para fins comparativos.

Portanto, o compressor SEG-Y-BZ2 é apresentado em cinco versões devido aos diferentes esquemas de reestruturação implementados. Destes, três foram propostos neste trabalho e dois foram extraídos da literatura. O desempenho das diferentes versões é comparado no decorrer deste capítulo, assim como o desempenho do PBZip2.

4.4 Métricas de Desempenho

Para avaliar o desempenho dos compressores nos experimentos realizados, quatro métricas são utilizadas: o tamanho do arquivo comprimido, o tempo de compressão, o tempo de descompressão e a redução no tamanho do arquivo comprimido. Estas métricas foram escolhidas, porque o objetivo deste trabalho é reduzir o tamanho do arquivo comprimido por meio das reestruturações propostas. Além disso, como os arquivos SEG-Y podem atingir Terabytes, o tempo de execução do compressor é considerado uma restrição e também deve ser reduzido.

A métrica que reflete o tamanho do arquivo comprimido é obtida pela razão entre o tamanho do arquivo SEG-Y comprimido (TC) e seu tamanho sem compressão

(TO). Ou seja, ela reflete o quanto se economizou de espaço através da compressão e sua fórmula é mostrada em na equação 4.1, em que V representa o valor da métrica em percentual.

$$V = (TC/TO) * 100 \tag{4.1}$$

As duas métricas de tempo (tempos de compressão e descompressão) são autoexplicativas e representam quão rápida é a execução dos programas enquanto realizam os processos de compressão e descompressão, respectivamente. Elas são computadas em segundos.

Por fim, a métrica que representa a redução do tamanho do arquivo comprimido é uma métrica que compara o tamanho do arquivo SEG-Y reestruturado e comprimido e o tamanho do arquivo SEG-Y comprimido em sua estrutura original. Ou seja, a métrica reflete o quanto de espaço se economizou devido a reestruturação. Ela é determinada pela razão entre o tamanho do arquivo SEG-Y comprimido por SEG-Y-BZ2 (TC_1) e o tamanho do arquivo SEG-Y comprimido por PBZip2 (TC_2). Sua fórmula é representada pela equação 4.2, em que R representa a redução alcançada.

$$R = (TC_1/TC_2) * 100 \tag{4.2}$$

Como os experimentos são realizados em sete arquivos SEG-Y que representam a bacia do Paraná e obtidos durante o mesmo processo de aquisição (ver seção 4.1), os valores das métricas exibido no decorrer deste capítulo corresponde à média aritmética dos resultados encontrados para cada um destes arquivos.

4.5 Otimização dos Parâmetros de SEG-Y-BZ2

Conforme apresentado na seção 3.2.3, o compressor SEG-Y-BZ2 possui três parâmetros de configuração que interferem diretamente no seu desempenho: o número de blocos em memória (*-b*), o número de traços por bloco (*-t*) e o número de *threads* de compressão/descompressão (*-n*). Destes, o número de traços por bloco é o único parâmetro a interferir no tamanho do arquivo comprimido, pois blocos com poucos traços podem ser expandidos de maneira a melhorar as taxas de compressão. Por outro lado, o número máximo de blocos em memória e, principalmente, o número de *threads* impactam nos tempos de compressão e descompressão.

Portanto, é preciso determinar o valor destes parâmetros de maneira a otimizar o desempenho do compressor, aumentando a taxa de compressão e reduzindo os tempos de compressão e descompressão. A escolha destes valores foi feita de maneira empírica, após cinco execuções em que um dos parâmetros assumiu valores pré-determinados, enquanto os demais permaneceram fixos. Para cada execução, o

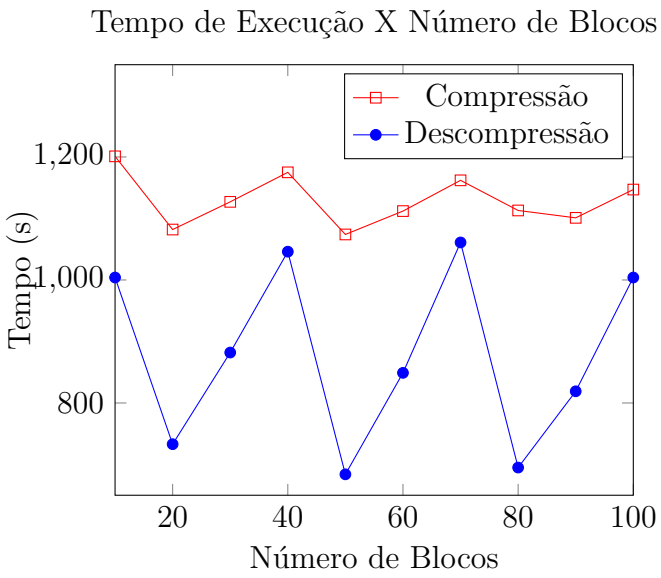


Figura 4.1: Tempo de execução do compressor SEG-Y-BZ2, medido em segundos, pela variação do número de blocos.

desempenho do compressor SEG-Y-BZ2 foi computado através das métricas apresentadas na seção anterior.

Como três estratégias de reestruturação de SEG-Y foram propostas, optou-se por apresentar os resultados apenas da segunda estratégia nestas execuções. Porém, todas as estratégias de reestruturação foram utilizadas na otimização do compressor e verificou-se que o valor dos parâmetros é o mesmo em todas elas. De fato, o desempenho do SEG-Y-BZ2 (em tempo e taxa de compressão) associado a cada reestruturação sofre pequena variação, uma vez que todas as estratégias recebem um vetor de amostras de 32 bits e fornecem três vetores com grupos de bits dessas amostras como saída.

O primeiro parâmetro estimado foi o número de blocos (*-b*). Para tal, fixou-se os demais parâmetros em 2000 traços por bloco e oito *threads* de compressão/descompressão, enquanto que o número de blocos em memória variou de 10 a 100 blocos, aumentando seu valor de 10 em 10 a cada nova execução. O objetivo deste experimento é determinar o número máximo de blocos em memória de maneira a reduzir o tempo ocioso das *threads* de compressão/descompressão, reduzindo assim o tempo de execução do compressor.

Como pode ser visto na Figura 4.1, o tempo de compressão variou entre 17 a 20 minutos, com média de 18 minutos e 40 segundos. A medida em que se aumentou o número de blocos em memória, o tempo de compressão oscilou, experimentando picos de 20 minutos e quedas a 18 minutos. Os dois menores tempos de compressão foram obtidos com 50 e 20 blocos em memória – 17 minutos e 54 segundos e 18 minutos e 2 segundos, respectivamente.

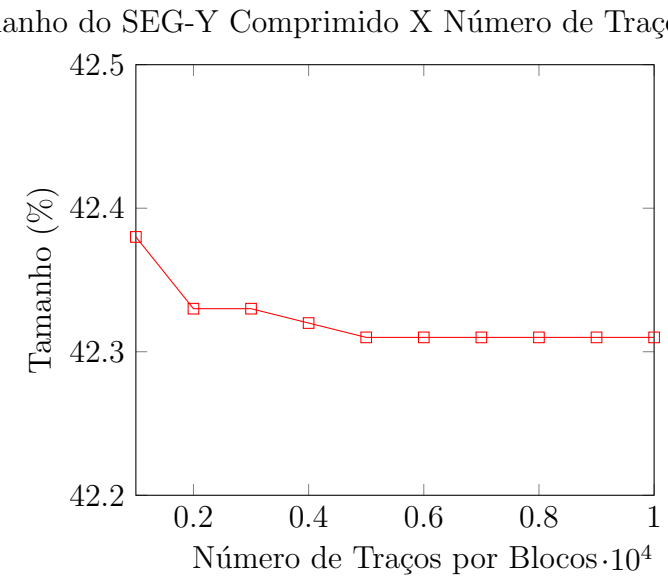


Figura 4.2: Percentual que o tamanho do arquivo SEG-Y comprimido por SEG-Y-BZ2 representa do tamanho do SEG-Y sem compressão, após variação do número de traços por bloco.

No que diz respeito ao tempo de descompressão, este sofreu variações mais bruscas que o tempo de compressão, assumindo valores entre 11 e 18 minutos, com tempo médio de 14 minutos. Contudo, o aumento e diminuição no tempo de descompressão ocorreram em conformidade com o aumento e diminuição no tempo de compressão. Neste caso, o menor tempo de descompressão foi alcançado com 50 e 80 blocos em memória – 11 minutos e 24 segundos e 11 minutos e 35 segundos, respectivamente.

Considerando o resultado deste experimento, concluiu-se que o melhor valor para o parâmetro *-b* é 50, ou seja, no máximo 50 blocos devem ser mantidos em memória para que seja possível reduzir o tempo de execução do compressor.

O segundo parâmetro estimado foi o número de traços por bloco (*-t*). Para tal, fixou-se o número de blocos em 50 e o número de *threads* de compressão/descompressão em oito, enquanto que o número de traços por bloco iniciou em 1000 e atingiu o valor máximo de 10000 traços, aumentado-se de 1000 em 1000 a cada nova execução.

Diferentemente dos outros experimentos de otimização dos parâmetros, para avaliar o número de traços por bloco foi computado o percentual que tamanho do arquivo SEG-Y comprimido representa do tamanho do SEG-Y sem compressão – com auxílio da equação 4.1. Neste caso, o objetivo é encontrar um valor para o parâmetro *-t* tal que o compressor reduza o tamanho do arquivo comprimido.

Os resultados mostraram que aumentar o número de traços por bloco produz uma pequena variação no tamanho do arquivo SEG-Y comprimido, conforme é mostrado na Figura 4.2. A diferença em comprimir arquivos SEG-Y com 1000 traços por bloco e 10000 traços por bloco é de apenas 0.07 pontos percentuals. Além disso, a maior

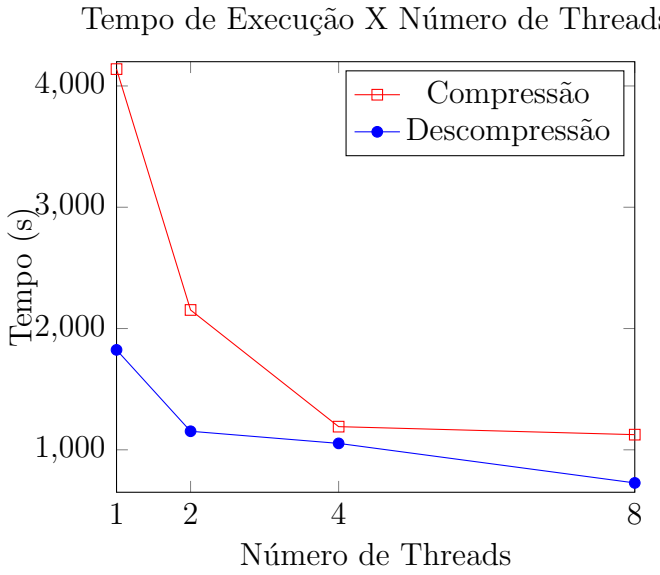


Figura 4.3: Tempo de execução do compressor SEG-Y-BZ2, medido em segundos, pela variação do número de threads.

variação ocorre quando o número de traços por bloco aumenta de 1000 para 2000 traços, reduzindo-se o tamanho do arquivo comprimido de 42.38% para 42.33% do tamanho do SEG-Y sem compressão. Como a variação é bem pequena no intervalo experimentado, optou-se por utilizar 2000 traços por bloco.

Por fim, o terceiro parâmetro estimado foi o número de *threads* de compressão/descompressão (*-n*). Neste experimento, o número de blocos foi fixado em 50 e o número de traços por bloco em 2000, enquanto que o número de *threads* assumiu os seguintes valores: 1, 2, 4 e 8 – sendo 8 a quantidade máxima de *cores* do processador Intel i7 empregado nos experimentos. Para avaliar o desempenho do compressor SEG-Y-BZ2, os tempos de compressão e descompressão foram computados.

Conforme pode ser visto na Figura 4.3, o número de *threads* de compressão/descompressão influencia significativamente o tempo de execução do compressor SEG-Y-BZ2. Durante o processo de compressão, é possível reduzir o tempo de execução de mais de 1 hora para pouco mais de 18 minutos, uma redução de 72.82%. Comportamento similar ao que foi encontrado no processo de descompressão, em que o tempo de execução foi reduzido de pouco mais de 30 minutos para cerca de 12 minutos, uma redução de 60.14%. Portanto, o valor ideal para o parâmetro *-n* é oito, o que habilita a criação de oito *threads* de compressão/descompressão.

Em resumo, os resultados apresentados no restante deste capítulo utilizam a seguinte configuração para o compressor SEG-Y-BZ2: máximo de 50 blocos em memória, 2000 traços por bloco e oito *threads* de compressão/descompressão. To-

davia, esta configuração é ideal para o ambiente de experimentação e conjunto de arquivos SEG-Y utilizados – e descrito na seção 4.2. Caso as configurações do ambiente sejam alteradas ou outros arquivos SEG-Y sejam utilizados, é possível que os valores destes parâmetros também sofram mudanças. Por exemplo, ao executar o compressor em um computador cujo processador tenha apenas dois núcleos de processamento, é possível que duas *threads* de compressão/descompressão proporcionem melhor desempenho do que oito *threads*.

4.6 Experimentos

Para melhor avaliar o desempenho do compressor SEG-Y-BZ2 e das diferentes versões de reestruturação, os arquivos SEG-Y cedidos passaram por um *workflow* básico de processamento, sendo comprimidos antes da execução do processamento e ao final de cada etapa. Como cada etapa de processamento altera as amostras do dado sísmico, a entropia do SEG-Y também varia, aumentando ou diminuindo as taxas de compressão. Desta forma, deseja-se mostrar que o compressor proposto possui melhor desempenho que o PBZip2, mesmo que as características do dado sísmico variem devido a diferentes estágios de processamento.

Para tal, o Seismic Unix[1] foi escolhido como ferramenta para execução do *workflow* de processamento. Ele é um programa em código aberto de pesquisa e processamento sísmico para ambientes Unix e é desenvolvido pelo *Center of Wave Phenomena* no Colorado, EUA. Além disso, o programa é executado em linha de comando, o que facilitou a criação de *shell scripts* para automatização dos experimentos.

Outro aspecto importante diz respeito ao formato de arquivo sísmico utilizado pelo Seismic Unix. Suas rotinas de processamento sísmico recebem como entrada e fornecem como saída um arquivo no formado "SU". Por essa razão, os arquivos SEG-Y foram convertidos para este formato e, ao final de cada etapa, o arquivo "SU" de saída foi convertido para SEG-Y que, por sua vez, foi comprimido e descomprimido. O processo de conversão entre formatos foi realizado com rotinas do próprio Seismic Unix.

O *workflow* de processamento escolhido contempla as seguintes etapas:

1. Correção da Distribuição Geométrica (DG);
2. Deconvolução (DC);
3. Mesclar traços (MT);
4. Análise de NMO (AN);

[1] `http://www.seismicunix.com/`

5. Stack (ST); e

6. Migração (MI).

O primeiro procedimento a ser executado é a ordenação dos traços do SEG-Y por *Common Depth Point* (CDP) que equivale a um ponto da superfície terrestre. Em seguida, ocorre a correção da distribuição geométrica (DG), responsável por atenuar a amplitude da onda sísmica; a deconvolução (DC); a mesclagem de traços para atenuação de ruídos (MT); e a análise de normal move-out (NMO). A etapa de stack (ST) é realizada a seguir, reduzindo drasticamente o tamanho do arquivo SEG-Y. Por fim, a etapa de migração (MI) corrige a posição dos eventos sísmicos.

Apesar de básico, este *workflow* foi escolhido, pois o objetivo deste trabalho não é processar o dado sísmico, mas mostrar que a compressão de SEG-Y é eficiente, independente do estágio de processamento em que o dado se encontra. Ademais, listar todos os *workflows* possíveis e escolher o melhor não é uma tarefa simples, tendo em vista a quantidade de etapas e inúmeras combinações de parâmetros. Além disso, este *workflow* de processamento é considerado representativo por conter etapas como redução de ruídos, atenuação de amplitudes, deconvolução, stack e migração – etapas que são frequentemente utilizadas por geólogos e geofísicos.

Ao todo, um arquivo SEG-Y é comprimido e descomprimido sete vezes nos experimentos realizados: uma vez quando o arquivo ainda não foi processado e uma vez ao final de cada uma das seis etapas do *workflow*. Os resultados destes experimentos são mostrados na seção seguinte.

4.7 Resultados

Após apresentar o ambiente de experimentação, os dados sísmicos, os compressores e as abordagens de reestruturação, as métricas de avaliação e, por fim, descrever os experimentos que foram executados, o restante deste capítulo se destina a apresentar e comentar os resultados obtidos. Estes resultados são apresentados em três seções, sendo a seção 4.7.1 responsável por comparar o resultado do compressor SEG-Y-BZ2 e das três abordagens de reestruturação propostas neste trabalho, elegendo a melhor delas. A seção 4.7.2 apresenta o desempenho do compressor SEG-Y-BZ2 utilizando duas estratégias de reestruturação encontrados na literatura: a diferença entre amostras consecutivas e uma separação de amostras proposta por Xie e Qin que foi adaptada para este trabalho. Por fim, a seção 4.7.3 compara o desempenho do compressor SEG-Y-BZ2 associado à melhor estratégia de reestruturação com o desempenho do PBZip2.

Conforme apresentado na seção 4.5, vale ressaltar que todos os experimentos descritos aqui utilizaram o SEG-Y-BZ2 configurado com parâmetros otimizados para

aumentar a taxa de compressão e reduzir os tempos de compressão e descompressão. O valor destes parâmetros são: máximo de 50 blocos em memória (-*b*), 2000 traços por bloco (-*t*) e oito *threads* de compressão/descompressão (-*n*).

Os gráficos que ilustram esta seção apresentam o desempenho alcançado nos experimentos. O eixo das ordenadas destes gráficos exibe o valor das métricas de desempenho, ou seja, tempos de compressão e descompressão, tamanho do arquivo comprimido ou redução no tamanho do arquivo obtida através da reestruturação. Por sua vez, o eixo das abscissas enumera as etapas de processamento na qual o desempenho foi computado, sendo composto por: SP (sem processamento), DG (correção da distribuição geométrica), DC (deconvolução), MT (mesclagem de traços), AN (análise de normal move-out), ST (stack) e MI (migração).

4.7.1 Desempenho das Estratégias de Reestruturação Propostas

O primeiro experimento envolveu o uso do compressor SEG-Y-BZ2 associado às abordagens de reestruturação propostas, computando seus desempenhos e comparando-os entre si. Desta forma, os gráficos que são mostrados no decorrer desta seção ilustram o desempenho de cada uma das estratégias de reestruturação e as identifica como "Versão 1", "Versão 2" e "Versão 3". Todas as versões separam os cabeçalhos das amostras, porém cada uma delas emprega uma estratégia de reestruturação diferente para separar os bits das amostras sísmicas.

A chamada "Versão 1" se refere ao compressor SEG-Y-BZ2 associado à reestruturação que separa as amostras do SEG-Y em 7 bits de expoente, 9 bits de sinal e mantissa alta, e 16 bits de mantissa baixa. Já a "Versão 2" emprega a reestruturação que separa as amostras sísmicas em 8 bits de sinal e expoente, 8 bits de mantissa alta e 16 bits de mantissa baixa. Por fim, a "Versão 3" reestrutura arquivos SEG-Y separando as amostras em 8 bits de sinal e expoente, 16 bits de mantissa alta e 8 bits de mantissa baixa. Portanto, cada i-ésima versão está relacionada à i-ésima estratégia de reestruturação proposta na seção 3.1.2 e ilustrada na Figura 3.5.

Também vale a pena ressaltar que as amostras sísmicas dos arquivos SEG-Y possuem arquitetura IBM, ilustrada na Figura 2.2, e que é formata por 1 bit de sinal, 7 bits de expoente e 24 bits de mantissa. As versões 1 e 2 dividem os 24 bits de mantissa em mantissas alta e baixa, sendo aquela formada pelos 8 bits mais significativos da mantissa e esta formada pelos 16 bits restantes. No caso da versão 3, a mantissa alta é formada pelos 16 bits mais significativos, enquanto que a mantissa baixa é formada pelos 8 bits restantes.

A Figura 4.4 ilustra o desempenho do SEG-Y-BZ2 e das três estratégias de reestruturação propostas, exibindo a evolução do tamanho do arquivo comprimido

em cada uma das etapas de processamento.

Observando esta figura, é possível notar que o SEG-Y sem processamento (SP) possui menor entropia que os arquivos processados em qualquer etapa e, ao ser comprimido, atinge no mínimo 40% do tamanho do SEG-Y sem compressão. Em contrapartida, as etapas DG, DC, ST e MI aumentam significativamente a entropia das amostras do SEG-Y e a compressão produz arquivos com 75 a 80% do tamanho do SEG-Y sem compressão. Todavia, a etapa de ST reduz significativamente o tamanho do SEG-Y (sem compressão), que passou de 44 Gigabytes (em média) para cerca de 100 Megabytes.

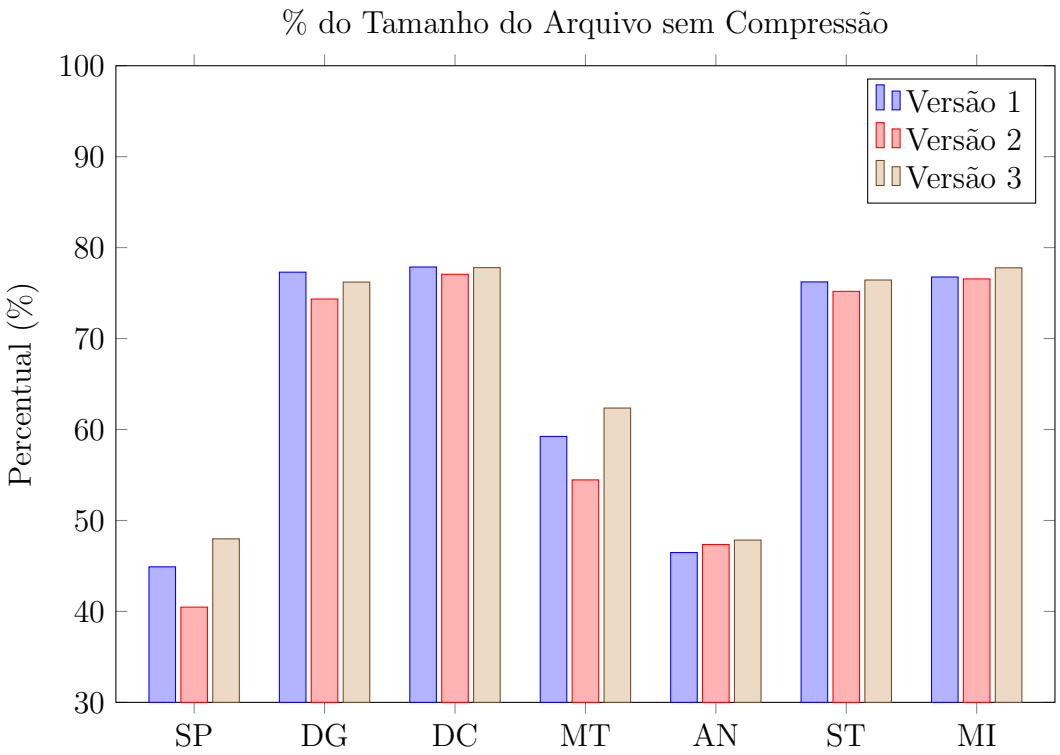


Figura 4.4: A figura mostra o percentual que o tamanho do SEG-Y comprimido representa do tamanho do arquivo sem compressão.

Comparando as três versões propostas, observa-se que o desempenho da compressão é bem similar nas etapas DG, DC, AN, ST e MI. Dentre estas, a maior diferença no tamanho do arquivo comprimido se deu na etapa DG com cerca de 3 pontos percentuais da versão 1 para versão 2 (de melhor desempenho) – com 77.30% e 74.35%, respectivamente. Por outro lado, as etapas SP e MT apresentaram diferenças mais acentuadas, com até 8 pontos percentuais de diferença entre a versão 2 e 3 na etapa MT – com 54.46% e 62.36%, respectivamente.

Além disso, no que tange ao tamanho do arquivo comprimido, a versão 2 do SEG-Y-BZ2 é a que apresenta melhor desempenho na maior parte das etapas de processamento, incluindo SP. A única etapa em que o desempenho da versão 2 foi superada foi a etapa AN e, ainda assim, a diferença entre versão 1 (menor arquivo

comprimido) e versão 2 é menor que 1 ponto percentual – 46.47% da versão 1 contra 47.35% da versão 2.

Quanto ao tempo de compressão exibido na Figura 4.5, nota-se que as diferentes versões de compressores demandam de 15 a 22 minutos para comprimir os arquivos SEG-Y. Além disso, o tempo de compressão varia de acordo com a entropia do arquivo, de maneira que quanto maior a entropia do arquivo SEG-Y, maior é o tempo de compressão. Por exemplo, ao comparar as etapas SP e DC, observa-se que o SEG-Y-BZ2 requer 15 minutos para comprimir o SEG-Y e obter um arquivo comprimido com 40% do tamanho do arquivo sem compressão na etapa SP. Já na etapa DC, o SEG-Y-BZ2 precisa de 22 minutos para comprimir o arquivo SEG-Y e produzir um arquivo comprimido com 77% do tamanho do SEG-Y sem compressão. Desta forma, o compressor gasta mais tempo e produz um arquivo comprimido de tamanho maior.

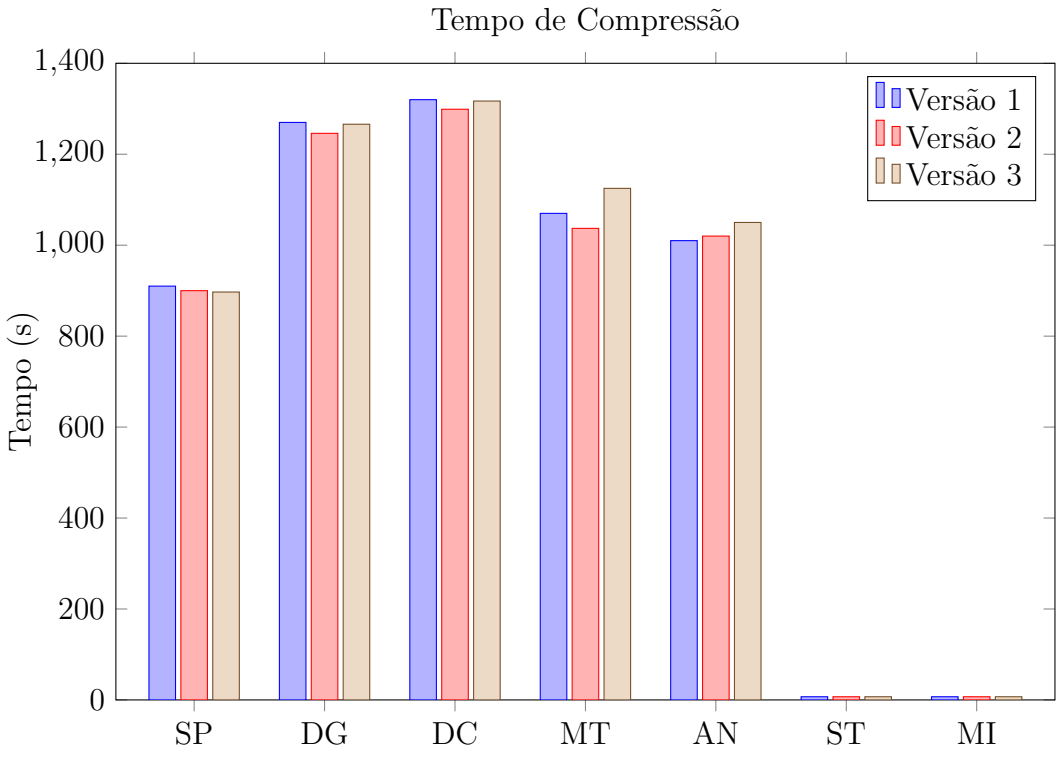


Figura 4.5: A figura mostra o tempo de compressão do SEG-Y-BZ2.

Também é possível notar uma redução expressiva no tempo de compressão de arquivos SEG-Y processados até as etapas ST e MI. Isto ocorre devido a redução drástica no tamanho do SEG-Y ocasionada pela etapa de stack. Como falado anteriormente, estes arquivos passaram de 44 Gigabytes para cerca de 100 Megabytes de tamanho. Desta forma, todas as versões do compressor necessitam em média de 7 segundos para processá-los.

A diferença de desempenho entre as três versões, no quesito tempo de compressão, é bem pequena. Porém, a versão 2 do compressor SEG-Y-BZ2 apresenta

menor tempo de compressão que as demais nas etapas de DG, DC e MT, além de demandar o mesmo tempo de compressão que as outras versões do compressor nas etapas de ST e MI.

Quanto ao tempo de descompressão, exibido na Figura 4.6, observa-se que o compressor gasta de 12 a 16 minutos para descomprimir e restaurar a estrutura original dos arquivos SEG-Y. Para as etapas ST e MI, em que os arquivos SEG-Y possuem tamanho médio de 100 Megabytes (sem compressão), todas as versões do compressor SEG-Y-BZ2 finalizaram a execução em 4 segundos.

Das sete etapas de processamento (incluindo SP, em que os arquivos SEG-Y não estão processados), a versão 1 do SEG-Y-BZ2 apresentou menor tempo de descompressão que as demais em AN, a versão 2 obteve melhor tempo nas etapas SP e DC, e a versão 3 requeriu menor tempo de descompressão que as demais na etapa DG. Quanto às etapas MT, ST e MI, o desempenho das três versões é praticamente o mesmo. Não obstante, nas etapas AN e DG em que as a versão 2 não foi a mais rápida, a diferença de tempo entre esta versão e a versão mais veloz é de apenas 14 e 27 segundos, respectivamente.

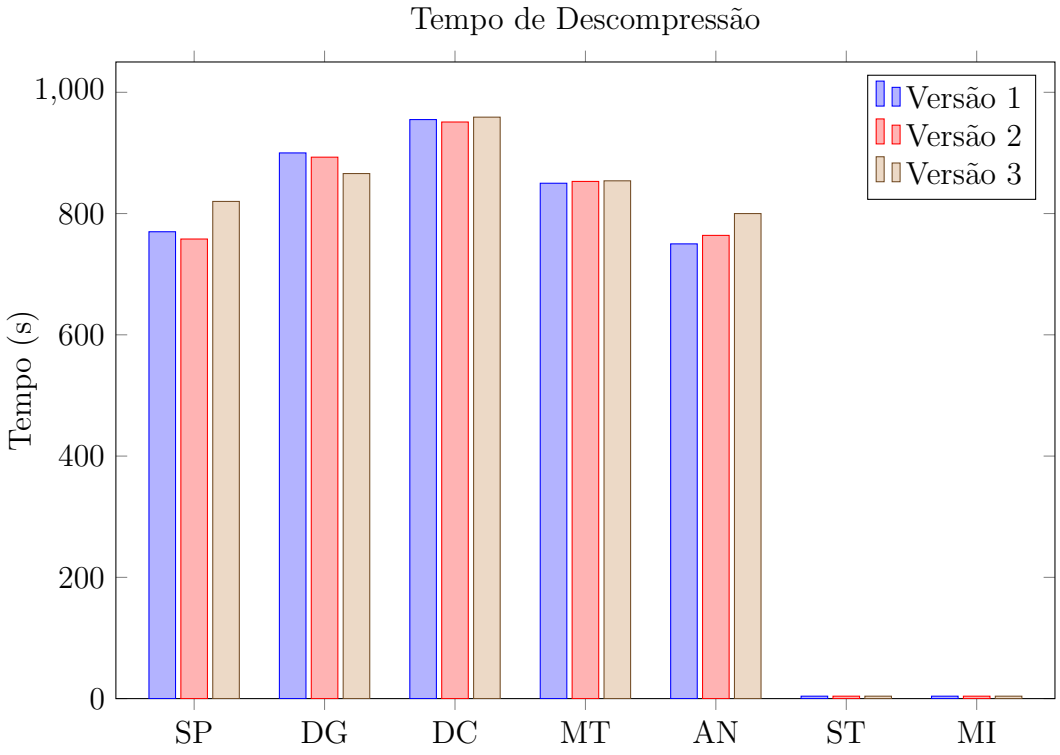


Figura 4.6: A figura mostra o tempo de descompressão do SEG-Y-BZ2.

Como a versão 2 do compressor SEG-Y-BZ2 obteve melhores taxas de compressão, produzindo arquivos SEG-Y comprimidos menores que as demais versões, e necessita de menor tempo de execução em boa parte das etapas de processamento, esta versão do compressor é considerada a mais eficiente. Ademais, nos casos em que a versão 2 foi superada em taxa de compressão ou tempo de compressão e des-

compressão, a diferença do seu desempenho para o desempenho da melhor versão é muito pequena.

Além disso, como a única diferença entre as três versões do compressor é a estratégia de reestruturação, conclui-se que a melhor reestruturação dentre as propostas para o conjunto de arquivos SEG-Y utilizados é aquela que separa as amostras em 8 bits de sinal e expoente, 8 bits de mantissa alta e 16 bits de mantissa baixa. Pois ela é capaz de reduzir a entropia dos arquivos SEG-Y de maneira a aumentar as taxas de compressão e reduzir o tempo de execução dos processos de compressão e descompressão. Porém, caso o conjunto de arquivos SEG-Y utilizados nos experimentos seja diferente, existe a possibilidade de que outra estratégia de reestruturação seja mais eficiente do que a encontrada aqui.

4.7.2 Desempenho das Estratégias de Reestruturação da Literatura

Nesta seção, são descritos os resultados dos experimentos que comparam o desempenho de três versões do compressor SEG-Y-BZ2. Uma que implementa a melhor estratégia de reestruturação proposta neste trabalho e identificada na seção 4.7.1, e duas versões que implementam estratégias de reestruturação identificadas na literatura.

Nas figuras que ilustram os desempenhos avaliados nesta seção, a legenda "Versão 2" refere-se ao compressor SEG-Y-BZ2 associado à reestruturação proposta, que separa as amostras do SEG-Y em 8 bits de sinal e expoente, 8 bits de mantissa alta e 16 bits de mantissa baixa. Esta estratégia de reestruturação é a melhor dentre aquelas que foram propostas neste trabalho, aumentando as taxas de compressão e reduzindo os tempos de compressão e descompressão.

Já a versão "Diff" refere-se ao compressor SEG-Y-BZ2 associado a diferença de amostras consecutivas. Esta abordagem de reestruturação foi descrita na literatura e consiste em representar as amostras através do resíduo da diferença entre amostras consecutivas de um mesmo traço. Além disso, este tipo de reestruturação é tão eficiente quanto à predição linear para amostras de pouca variação, porém requer menor tempo de execução, uma vez que os coeficientes utilizados na diferença são pré-determinados.

Vale a pena ressaltar que a fórmula para o cálculo da diferença entre amostras implementada consiste em subtrair uma amostra da amostra subsequente, no mesmo traço. Desta forma, a primeira a mostra do traço é representada pelo seu valor (sem alteração), enquanto que as demais amostras são representadas pelo resultado da diferença entre o seu valor e o valor da amostra seguinte. Para utilizar uma fórmula que envolva diversas amostras do traço é preciso ter conhecimento prévio do dado

sísmico ou então aderir a uma abordagem dinâmica de cálculo, de maneira a minimizar os efeitos da variação das amostras. Por outro lado, utilizar uma abordagem simples e estática como a subtração de amostras consecutivas produz bons resultados em dados sísmicos que não sofrem variações demasiadas, que é o caso dos dados sísmicos utilizados nestes experimentos.

Por fim, a versão "Xie e Qin" do compressor SEG-Y-BZ2 está associada à estratégia de reestruturação proposta por Xie e Qin em [27]. No esquema de compressão deles, os bits das amostras e da diferença das amostras são divididos em sinal, expoente e mantissa, e utilizados para decidir entre comprimir os bits das amostras ou das diferenças de amostras. Independente da decisão tomada pelo algoritmo, sinal, expoente e mantissa são comprimidos separadamente com *Arithmetic Coding*. Desta forma, esta abordagem foi simplificada e adaptada pelo SEG-Y-BZ2 para juntar o bit de sinal e os 7 bits expoente das amostras em um bloco e, em outro bloco, armazenar todos os 24 bits da mantissa das amostras. Ao final da reestruturação, os blocos (de sinal e expoente, e da mantissa) são comprimidos separadamente.

A alteração na proposta de Xie e Qin foi realizada, pois dos 32 bits da amostra, a parte que mais impacta a compressão é composta pelos bits da mantissa, que neste caso são mantidos conforme a proposta original. A separação do bit de sinal é mais vantajosa quando há longas sequência de bits repetidos, o que torna possível a utilização do algoritmo *Run-Length Encoding*. Desta forma, deseja-se mostrar que a separação da mantissa em duas partes com bits mais e menos significativos constitue uma melhor estratégia do que comprimir todos os bits juntos.

Conforme pode ser observado na Figura 4.7, a estratégia de reestruturação proposta neste trabalho possui as maiores taxas de compressão nas seis etapas do *workflow* de processamento sísmico (da etapa DG à MI). Desta forma, o compressor SEG-Y-BZ2 que implementa esta abordagem de reestruturação ("Versão 2") produziu os menores arquivos comprimidos. A única etapa em que a "Versão 2" do compressor não obteve a maior taxa de compressão foi a etapa SP. Nela, a diferença entre amostras ("Diff") produziu um arquivo comprimido com 40.03% do tamanho do SEG-Y sem compressão, contra 40.47% da "Versão 2" – uma diferença de 0.44 pontos percentuais.

Nota-se ainda que a diferença entre amostras ("Diff") apresenta o pior desempenho em cinco etapas de processamento (DG, DC, AN, ST e MI), chegando a produzir arquivos comprimidos com 89% do tamanho do SEG-Y (sem compressão) na etapa DC. Em contrapartida, tanto a estratégia de reestruturação proposta aqui ("Versão 2") quanto aquela proposta por Xie e Qin ("Xie e Qin") produziram arquivos comprimidos com cerca de 77 a 78% do tamanho do SEG-Y (sem comapctação) nesta etapa – uma diferença de mais de 10 pontos percentuais para a diferença entre amostras ("Diff").

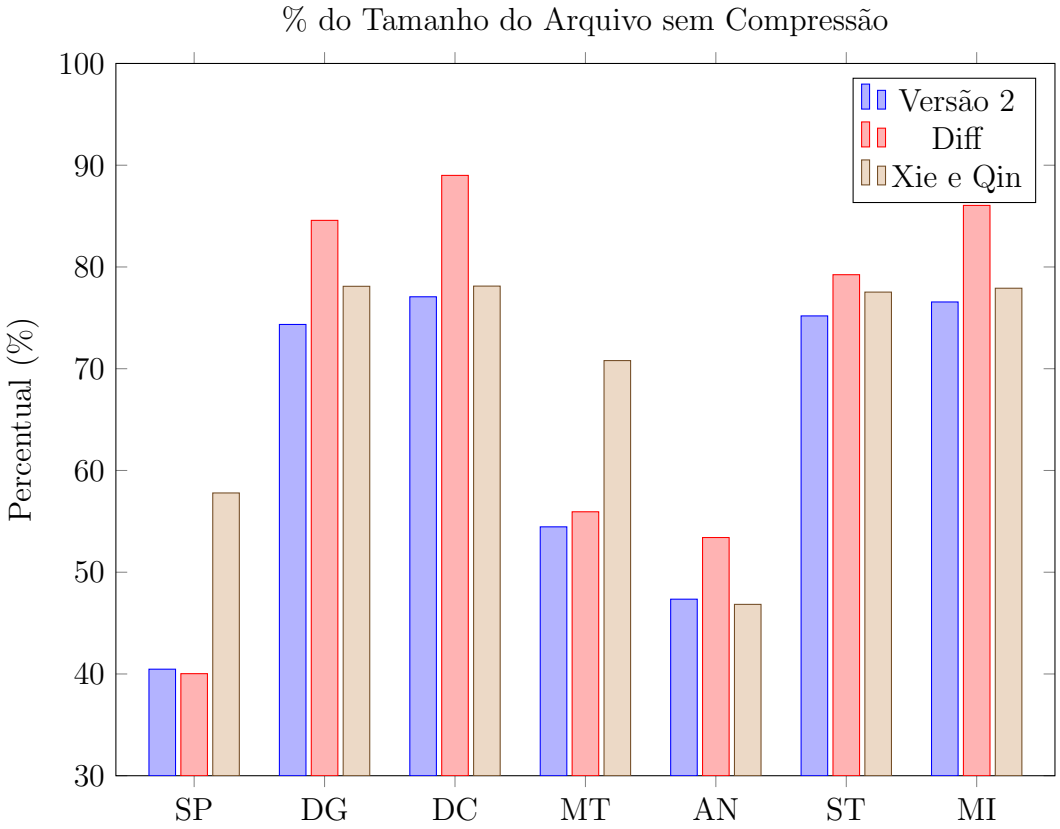


Figura 4.7: A figura mostra o percentual que o tamanho do SEG-Y comprimido representa do tamanho do arquivo sem compressão.

Todavia, a abordagem proposta por Xie e Qin (“Xie e Qin”) também apresentou comportamento similar a diferença entre amostras (“Diff”), porém, apenas na etapa MT. Neste caso, a estratégia de Xie e Qin produziu um arquivo comprimido com 70.80% do tamanho do SEG-Y sem compressão, enquanto que as outras versões apresentaram arquivos com 54 a 55% do tamanho do SEG-Y. Uma diferença superior a 15 pontos percentuais, o que pode significar muitos Gigabytes a mais no tamanho do arquivo comprimido.

Quanto ao tempo de compressão exibido na Figura 4.8, o melhor desempenho ficou a cargo da diferença entre amostras (“Diff”), seguida pelo SEG-Y-BZ2 associado a melhor reorganização proposta neste trabalho (“Versão 2”) e, por último, da estratégia proposta por Xie e Qin (“Xie e Qin”). De fato, a diferença entre amostras é um processo de reestruturação mais simples, pois não requer a criação de vetores temporárias para armazenar a diferença, além de realizar apenas a subtração cujo resultado é armazenado no próprio vetor de amostras. Por outro lado, tanto a “Versão 2”, quanto à “Xie e Qin” são estratégias de reestruturação que copiam partes das amostras para diferentes vetores de dados que serão comprimidos ao final do processo.

A diferença no tempo de compressão entre a “Versão 2” e a diferença entre

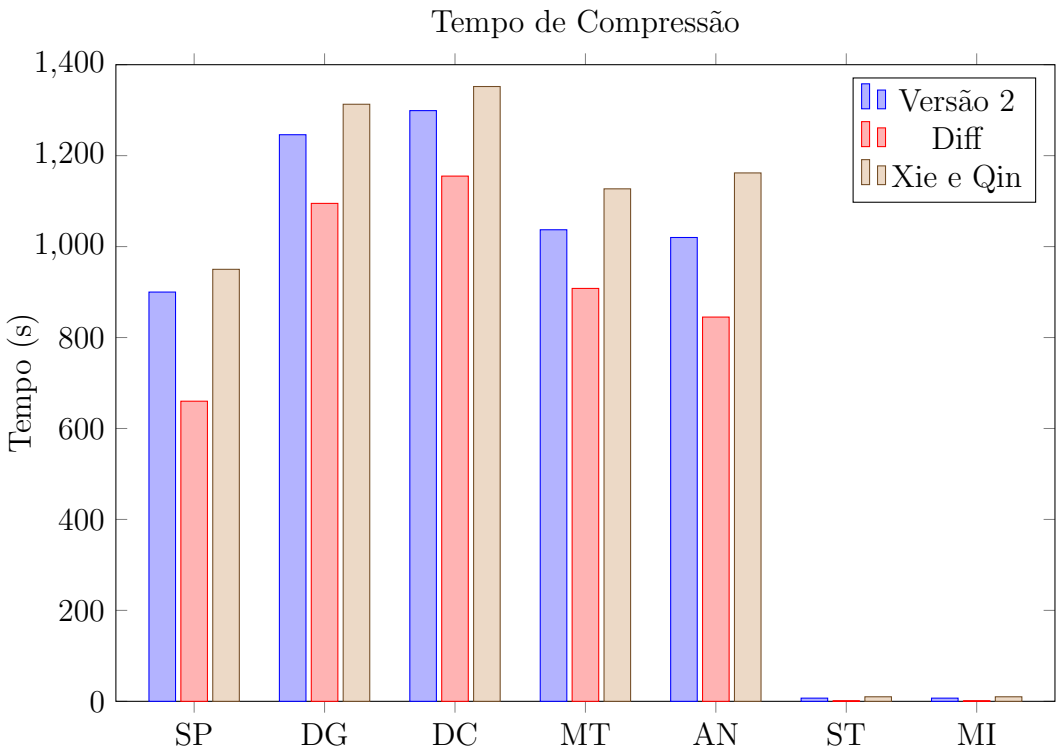


Figura 4.8: A figura mostra o tempo de compressão do SEG-Y-BZ2.

amostras ("Diff") é de 2 a 4 minutos nas etapas SP, DG, DC, MT e AN. Já nas etapas ST e MI, em que os arquivos SEG-Y ficam com 100 Megabytes em média, o tempo de compressão para a diferença entre amostras ("Diff") é de 1 segundo contra 7 segundos da "Versão 2". Apesar da diferença entre amostras ser a reestruturação mais rápida, ela paga um preço alto, porque produz arquivos comprimidos de maior tamanho na maior parte dos casos experimentados – conforme mostrado na Figura 4.7.

Analisando o tempo de descompressão exibido na Figura 4.9, nota-se que a diferença entre amostras ("Diff") não é mais unanimidade. Nas etapas de DG, DC e AN, esta estratégia foi a que demandou maior tempo de descompressão, enquanto que na etapa MT o maior tempo de descompressão foi alcançado pela reestruturação proposta por Xie e Qin ("Xie e Qin"). No que tange o SEG-Y-BZ2 associado à melhor reestruturação proposta neste trabalho ("Versão 2"), esta abordagem alcançou os melhores tempos de descompressão nas etapas DG, DC, AN, ST e MI, mesmo que com uma diferença muito pequena. Porém, em nenhum momento a "Versão 2" obteve o pior tempo de descompressão.

Uma causa possível para que a diferença entre amostras ("Diff") tenha maior tempo de descompressão, pode ser esclarecida ao comparar o tamanho do arquivo comprimido (Figura 4.7) e o tempo de descompressão (Figura 4.9). Justamente nas etapas de DG, DC e AN em que a diferença entre amostras obteve o maior tempo de descompressão, o processo de compressão produziu arquivos maiores do que as

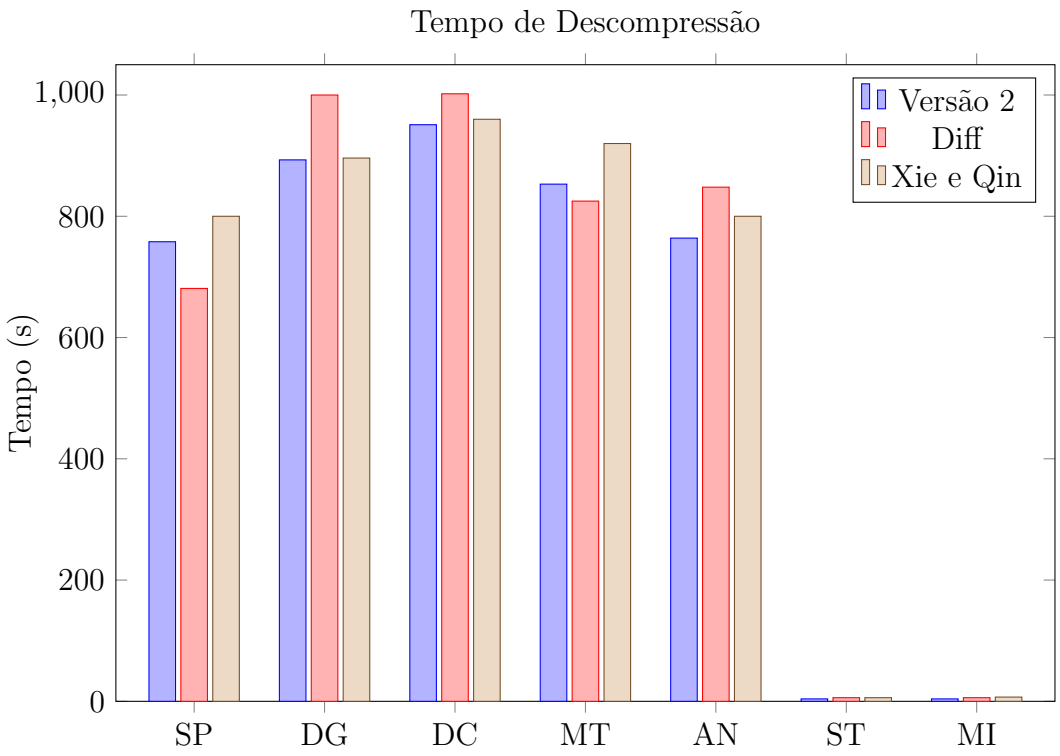


Figura 4.9: A figura mostra o tempo de descompressão do SEG-Y-BZ2.

demais versões experimentadas. Esta ocorrência também pode ser observada com o SEG-Y-BZ2 associado ao esquema de reestruturação proposto por Xie e Qin ("Xie e Qin") na etapa de MT. Quanto às etapas ST e MI, apesar da diferença entre amostras ("Diff") ter produzido os maiores arquivos comprimidos, a diferença no tempo de descompressão não é tão visível, visto que os arquivos são muito pequenos – poucos Megabytes.

Em resumo, esta seção mostrou que a melhor estratégia de reestruturação proposta neste trabalho – que divide as amostras em sinal e expoente (8 bits), mantissa alta (8 bits) e mantissa baixa (16 bits) – superou as duas estratégias de reestruturação experimentadas e que foram selecionadas na literatura. Todavia, apesar da reestruturação proposta perder em tempo de compressão para a diferença entre amostras consecutivas, esta perde na taxa de compressão e no tempo de descompressão, sendo este último influenciado principalmente pelo tamanho do arquivo comprimido. Já a reestruturação adaptada de Xie e Qin obteve desempenho comparável à reestruturação proposta no que diz respeito às taxas de compressão, mas a compressão conjunta dos 24 bits da mantissa se mostrou pior do que a separação da mantissa em grupos de bits mais e menos significativos para alguns arquivos SEG-Y processados.

4.7.3 Desempenho dos Compressores SEG-Y-BZ2 e PBZip2

Para finalizar os experimentos, esta seção compara o desempenho do compressor SEG-Y-BZ2 associado ao melhor esquema de reestruturação proposto – que separa amostras do SEG-Y em sinal e expoente (8 bits), mantissa alta (8 bits) e mantissa baixa (16 bits) – e o desempenho do compressor PBZip2. Ambos os compressores realizam os mesmos processos de compressão e descompressão, empregando algoritmos como a transformada *Burrow-Wheeler* e *Huffman*. A diferença ocorre no emprego da reestruturação de arquivos SEG-Y por parte do compressor SEG-Y-BZ2 para redução da entropia do dado sísmico, enquanto que o PBZip2 apenas comprime tais arquivos na forma em que estão representado.

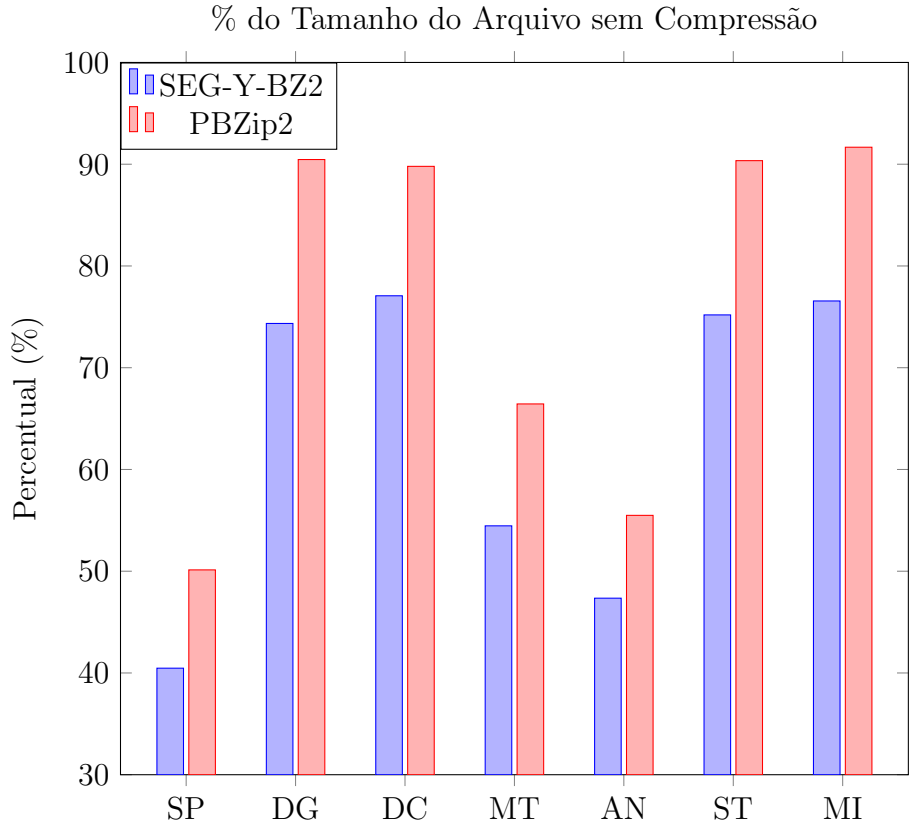


Figura 4.10: A figura mostra o percentual que o tamanho do SEG-Y comprimido representa do tamanho do arquivo sem compressão.

Conforme pode ser observado na Figura 4.10, o compressor SEG-Y-BZ2 possui maior taxa de compressão do que o PBZip2 em todas as etapas do *workflow* de processamento (incluindo a etapa SP, em que os arquivos SEG-Y não foram processados). Desta forma, ele é capaz de produzir arquivos comprimidos de menor tamanho do que o PBZip2. E a diferença no tamanho não é pequena, podendo variar de 8 a 15 pontos percentuais. Isso significa que, enquanto o SEG-Y-BZ2 gera um arquivo SEG-Y comprimido com 33 Gigabytes, o compressor PBZip2 gera um

arquivo comprimido com 40 Gigabytes, uma diferença de 7 Gigabytes – considerando os valores da etapa DG e que um arquivo SEG-Y tem tamanho médio de 44 Gigabytes nesta etapa.

Mesmo nas etapas de ST e MI, em que os arquivos SEG-Y têm tamanho muito menor (cerca de 100 Megabytes), a diferença nas taxas de compressão é expressiva e não pode ser ignorada. Pois, por mais que um arquivo SEG-Y tenha apenas 100 Megabytes, é possível que empresas de exploração e análise sísmica gerem centenas ou milhares desses arquivos, variando técnicas e parâmetros do processamento. Desta forma, o grande número de arquivos SEG-Y processados até as etapas ST e MI pode ser tão numeroso que o espaço economizado pela utilização do SEG-Y-BZ2 seja significativo.

A Figura 4.11 exibe o percentual de redução no tamanho do arquivo comprimido alcançado pelo SEG-Y-BZ2 em relação ao PBZip2. Esta redução ocorre devido ao uso da reestruturação, tendo em vista que os algoritmos de compressão são os mesmos para ambos os compressores. Conforme pode ser notado, é possível reduzir o tamanho do arquivo comprimido de 14 a 20% com o auxílio da reestruturação.

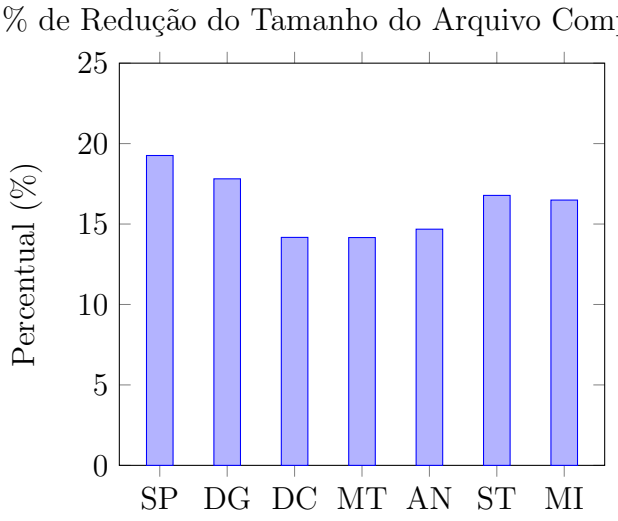


Figura 4.11: A figura mostra o percentual de redução no tamanho do arquivo comprimido com SEG-Y-BZ2 em relação ao arquivo comprimido por PBZip2.

Nos experimentos, os arquivos sem processamento (etapa SP) foram aqueles que, após reestruturação e compressão, obtiveram maior redução no tamanho do arquivo comprimido. Estes arquivos passaram de 22 Gigabytes para 17.8 Gigabytes, uma redução de quase 20% – considerando que os arquivos SEG-Y utilizados nos experimentos tem 44 Gigabytes em média. Em outras etapas a redução é um pouco menor que 20% (cerca de 14 a 17%), entretanto a economia em termos de espaço pode ser muito maior, como é o caso da etapa DG em que a redução foi de 17.81% e o arquivo comprimido passou de 40 Gigabytes para 33 Gigabytes, conforme relatado anteriormente.

Apesar dos compressores implementarem os mesmos algoritmos de compressão e possuirem oito *threads* de compressão/descompressão, seus tempos de compressão são bem diferentes, conforme apresentado na Figura 4.12. É possível observar uma diferença significativa no tempo de compressão de arquivos SEG-Y processados até a etapa DG e DC. Enquanto o SEG-Y-BZ2 comprimiu estes arquivos em cerca de 21 minutos, o PBZip2 precisou de pouco mais de 25 minutos para completar a tarefa.

Comparando a taxa de compressão destes algoritmos (Figura 4.10) e o tempo de compressão (Figura 4.12), observa-se que o compressor PBZip2 produziu os arquivos comprimidos com 90% do tamanho do SEG-Y (sem compressão) das etapas DG, DC, ST e MI. E justamente nas etapas DG e DC, ele alcançou o maior tempo de compressão, o que leva a crer que o tempo de compressão dos arquivos SEG-Y é muito maior do que o tempo de reestruturação e que o processo de reestruturação além de aumentar as taxas de compressão, também reduz o tempo de compressão.

Quanto às etapas ST e MI, como os arquivos SEG-Y são muito pequenos (cerca de 100 Megabytes), o tempo gasto pelo SEG-Y-BZ2 na reestruturação do arquivo é bem maior do que o tempo gasto na compressão. Por essa razão, o PBZip2 obteve menores tempos de compressão e uma vantagem de apenas 3 segundos.

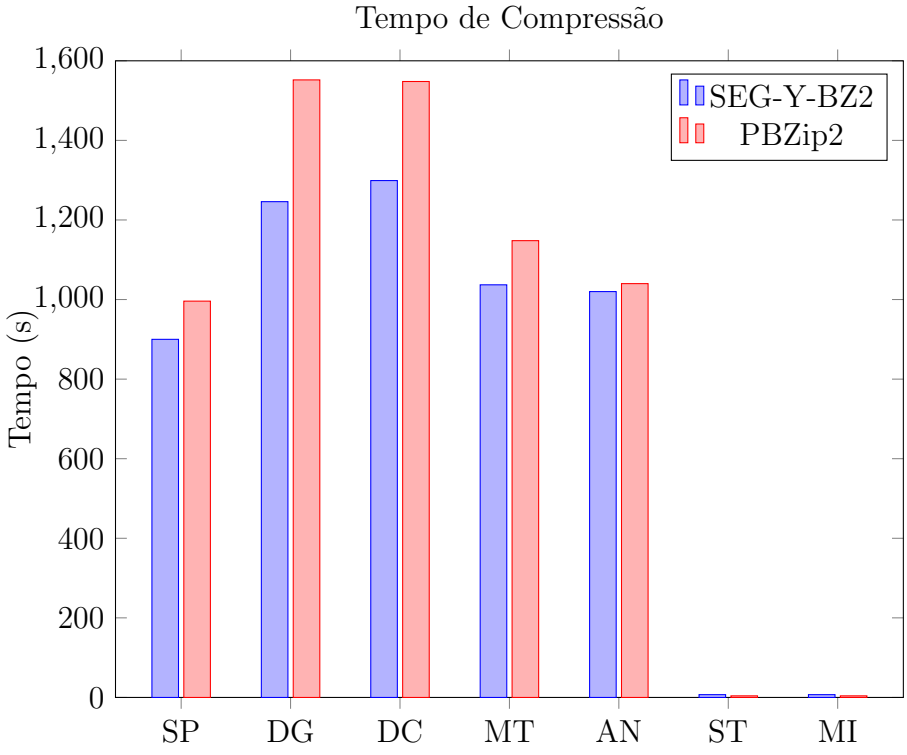


Figura 4.12: A figura mostra o tempo de compressão do SEG-Y-BZ2 e PBZip2.

O tempo de descompressão, mostrado na Figura 4.13, também foi favorável ao SEG-Y-BZ2 nas etapas SP, DG, DC, MT e AN, por ter produzido os menores arquivos comprimidos. A maior diferença neste quesito também ocorreu nas etapas DG e DC. Na primeira, o SEG-Y-BZ2 necessitou em média de 14 minutos e 53

segundos na descompressão dos arquivos SEG-Y, enquanto que o PBZip2 precisou de 17 minutos e 24 segundos para finalizar a operação. Já na etapa DC, o SEG-Y-BZ2 descomprimiu os arquivos em 15 minutos e 51 segundos, enquanto que o PBZip2 o fez em 17 minutos e 43 segundos.

Da mesma forma como ocorreu com o tempo de compressão, o tempo de descompressão do PBZip2 nas etapas ST e MI foi inferior ao tempo do SEG-Y-BZ2. Mais uma vez, o arquivo SEG-Y é pequeno demais e, neste caso, o tempo da reestruturação é superior ao tempo da descompressão.

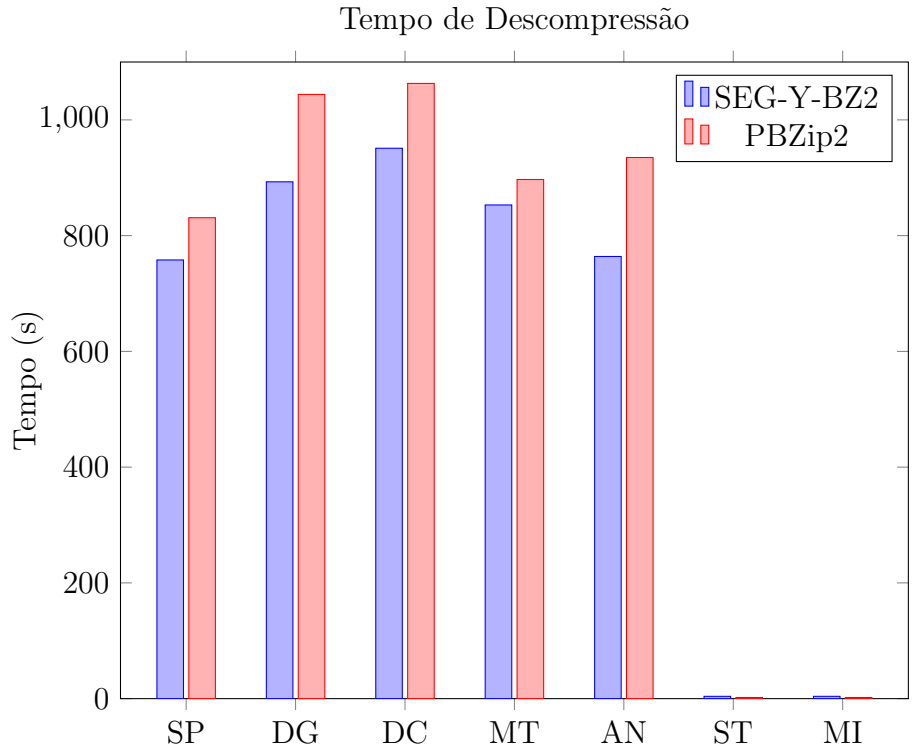


Figura 4.13: A figura mostra o tempo de descompressão requerido por SEG-Y-BZ2 e PBZip2.

Em resumo, esta seção comprovou que o compressor SEG-Y-BZ2, que implementa o melhor esquema de reestruturação – separando as amostras do SEG-Y em sinal e expoente (8 bits), mantissa alta (8 bits) e mantissa baixa (16 bits) –, possui melhor desempenho que o compressor PBZip2. Apesar de ambos os compressores implementarem os mesmos algoritmos de compressão e executarem suas operações com oito *threads* de compressão/descompressão, o SEG-Y-BZ2 com sua reestruturação de arquivos SEG-Y foi capaz de reduzir o tamanho do arquivo comprimido em até 20%, em comparação com o PBZip2. Além disso, ele também apresentou melhores tempos de compressão e descompressão, pois o processo de compressão e descompressão dos arquivos SEG-Y em sua estrutura original se mostrou mais custoso do que o processo de reestruturação destes arquivos. Portanto, a reestruturação do arquivo SEG-Y foi capaz de aumentar as taxas de compressão e também reduzir

os tempos de compressão e descompressão, uma vez que a entropia do dado diminuiu e, consequentemente, o tamanho do arquivo comprimido.

Capítulo 5

Conclusões

Apesar dos avanços tecnológicos em termos de armazenamento e transmissão de dados, o grande volume de dados sísmicos compõe um desafio para organizações cujas atividades envolvem a exploração e análise sísmica, mas principalmente a extração de hidrocarbonetos. Devido ao emprego crescente de técnicas de aquisição de dados sísmicos mais avançadas – de três e quatro dimensões –, o volume deste dado, que representa a estrutura geológica da superfície terrestre, tem ultrapassado a ordem de Gigabytes, atingindo até mesmo Terabytes. Desta forma, o crescente aumento na capacidade de armazenamento de *storages* e velocidade de transmissão de redes não são suficientes para contornar as dificuldades na manipulação destes dados.

Para superar as dificuldades, as organizações recorrem cada vez mais a técnicas de compressão que buscam identificar padrões repetidos de símbolos que compõem o dado e substituí-los por sequências menores, reduzindo assim seu volume. Entretanto, muitas técnicas descritas na literatura possuem baixo desempenho quando aplicadas em dados sísmicos, devido à grande variação das amostras. Por essa razão, é comum o emprego de abordagens que reordenam (ou reestruturam) os símbolos que compõem o dado sísmico, visando reduzir sua entropia e, consequentemente, aumentar a eficiência das técnicas de compressão.

Este trabalho propôs, entre outras coisas, uma taxonomia para organizar as diferentes técnicas de compressão de dados sísmicos descritas na literatura, pois nenhum trabalho relacionado a este tópico foi encontrado. Ela identifica dois grandes grupos de técnicas de compressão: sem perdas e com perdas. O primeiro grupo é formado por técnicas de compressão que não promovem alterações no dado sísmico após execução dos processos de compressão e descompressão. Enquanto que o segundo grupo é composto por técnicas de compressão que descartam parte do dado para atingir a compressão, inserindo ruídos no processo de descompressão. Além disso, cada um destes grupos foi dividido em grupos menores cujo objetivo é especializar, organizar e melhor classificar as diversas técnicas encontradas. Assim, as técnicas de compressão sem perdas podem ser aplicadas nas amostras em sua estrutura ori-

ginal ou em amostras reestruturadas, enquanto que a maior parte das técnicas de compressão com perdas se resume à aplicação de transformadas, quantização (ou descarte de parte dos dados) e codificação.

Após a organização da literatura relacionada, foi possível observar um maior número de pesquisas voltadas para a compressão com perdas, mesmo sabendo que o emprego destas técnicas sofre resistência por parte de geofísicos e geólogos que desejam manter inalteradas as características do dado . Por essa razão e somado aos poucos trabalhos relacionados à compressão sem perdas, o restante deste trabalho focou em oferecer uma alternativa simples, eficaz e de baixo custo monetário para este tipo de compressão, que consiste em utilizar os compressores de dados genéricos existentes no mercado.

São muitos os compressores existentes no mercado que são gratuitos e podem ser adquiridos prontamente via Internet, como BZip2, PBZip2, 7Zip, entre outros. Contudo, eles são destinados à compressão de dados genéricos (fotos, vídeos, texto, etc) e seu desempenho pode ser ineficaz quando o assunto é a compressão de dados sísmicos. A solução para contornar esta dificuldade é a adoção de uma estratégia de reestruturação do dado sísmico cujos objetivos são aumentar as taxas de compressão e reduzir o tempo de compressão e descompressão destas ferramentas.

Desta forma, foram propostas três abordagens para reestruturação do dado sísmico, representados no formato SEG-Y, que se assemelham em comprimir cabeçalhos (textuais e binários) separados das amostras sísmicas cuja arquitetura é IBM. Por outro lado, elas se diferenciam quanto à forma de reestruturar as amostras em ponto flutuante, separado-as em diferentes partes que devem ser agrupadas e comprimidas independentes umas das outras. Juntando a reestruturação com os compressores existentes, foi possível tornar o processo de compressão e descompressão muito mais eficiente.

Como prova de conceito, foi desenvolvido um compressor, *multithread*, que divide um arquivo SEG-Y em blocos, reorganiza-os e comprime-os independentemente. Ele utiliza a biblioteca *libbzip2* que fornece as rotinas de compressão de dados em memória, as mesmas implementadas nos compressores BZip2 e PBZip2, este último considerado o mais eficiente em tempo de compressão e descompressão nos experimentos realizados. Desta forma, o objetivo com o desenvolvimento do SEG-Y-BZ2 é mostrar que o processo de reestruturação integrado com os compressores já existentes é eficiente na compressão de dados sísmicos, representados no formato SEG-Y.

Já que o compressor SEG-Y-BZ2 utiliza os mesmos algoritmos de compressão que o PBZip2, foi possível comparar seus desempenhos nos diversos experimentos realizados. Os resultados mostraram que o processo de reestruturação do arquivo SEG-Y é eficaz, reduzindo a entropia do arquivo, o que melhora as taxas de compressão e inclusive diminui os tempos de compressão e descompressão, uma vez que,

com a redução da entropia do dado, as árvores de *Huffman* tem tamanho menor e, consequentemente, gasta-se menos tempo na codificação. Conforme foi atestado, o SEG-Y-BZ2 foi capaz de reduzir o tamanho do arquivo comprimido em até 20% em relação ao PBZip2. Esta redução variou de 15% a 20% aproximadamente, dependendo da etapa de processamento em que o arquivo SEG-Y se encontrava. Todavia, em todas as etapas de processamento experimentadas, foi possível obter arquivos SEG-Y comprimidos de menor tamanho.

Não obstante, o compressor desenvolvido alcançou os menores tempos de compressão, uma vez que os processos de reestruturação e compressão em conjunto demandam menor tempo de execução do que a compressão aplicada sobre o SEG-Y em sua estrutura original. Isso ocorre devido à reestruturação que criou blocos de menor entropia e, consequentemente, mais simples de serem comprimidos. Ademais, o SEG-Y-BZ2 também apresentou menor tempo de descompressão, superando o compressor mais eficiente experimentado – o PBZip2. A redução no tempo de descompressão ocorre, principalmente, porque foi possível reduzir o tamanho do arquivo comprimido significativamente.

Quanto aos esquemas de reestruturação propostos, aquele que obteve o melhor desempenho separava os cabeçalhos e amostras do arquivo SEG-Y e comprimia-os separadamente. Além disso, para cada bloco do arquivo SEG-Y, as amostras sísmicas, cuja arquitetura é IBM e possuem 32 bits, foram separadas em sinal e expoente (8 bits), mantissa alta (8 bits) e mantissa baixa (16 bits). Estas partes foram agrupadas em três blocos diferentes que, por fim, foram comprimidos separadamente. Assim, para cada bloco do arquivo SEG-Y, o SEG-Y-BZ2 comprime quatro blocos independentemente: um de cabeçalhos, um de sinal e expoente, um de mantissa alta e um de mantissa baixa.

Este esquema de reestruturação se mostrou muito mais eficaz que os demais propostos, além de superar abordagens presentes na literatura que propõem a diferença de amostras de um mesmo traço sísmico e outra que propõe que a amostra sísmica seja divida em sinal, expoente e mantissa. Os experimentos também utilizaram as rotinas de compressão da biblioteca *libbzip2* e os resultados mostraram que a reestruturação proposta possibilita maiores taxas de compressão e bons tempos de descompressão. Contudo, a diferença entre amostras é a reestruturação que possibilita o menor tempo de compressão, tendo em vista que não é preciso gerenciar estruturas de memória auxiliar nem realizar cópia dos bits das amostras para diferentes vetores, durante a compressão.

Portanto, conclui-se que os compressores existentes no mercado constituem uma opção simples e de baixo custo para compressão de dados sísmicos, mas que precisam aumentar suas taxas de compressão e reduzir os tempos de compressão e descompressão. Uma maneira eficiente de fazê-lo, sem ter que alterar os algoritmos

de compressão é aplicar uma estratégia de reestruturação com o intuito de reduzir a entropia do dado sísmico, como foi mostrado neste trabalho. Além disso, como o tempo é uma restrição das aplicações sísmicas, é possível integrar a estratégia de reestruturação e o compressor em um programa *multithread*, tornando possível a redução nos tempos de execução.

Todavia, é preciso pesquisar novas formas de reestruturação do dado sísmico que levem em conta as características deste tipo de dado. Inclusive, é possível utilizar os próprios atributos dos cabeçalhos dos traços do SEG-Y para ordena-los ou agrupá-los, de maneira a reduzir entropia do arquivo antes da compressão. É provável que uma abordagem de reestruturação que considere atributos do dado sísmico, como fonte geradora, amplitude, frequência de aquisição, ou até mesmo as características da superfície em questão, possam produzir melhores resultados do que as técnicas de compressão sem perdas existentes atualmente.

Além disso, a combinação de estratégias de reestruturação pode vir a ser uma interessante alternativa do ponto de vista de taxas de compressão. Talvez, utilizar duas ou mais abordagens de ordenação e reestruturação dos dados sísmicos reduza as taxas de compressão. Porém, quanto maior ou mais complexo é este pré-processamento, maior será o tempo de compressão e também de descompressão, visto que é preciso reconstruir a estrutura original do dado.

Outra possibilidade é investigar o desempenho de diferentes estratégias de reestruturação de dados sísmicos em diversas etapas do *workflow* de processamento – não apenas nas etapas utilizadas neste trabalho. Neste caso, é preciso verificar se existem estratégias que se adequam melhor a dados sísmicos que foram processados por uma etapa, ou por um conjunto de etapas, ou ainda não atingiram determinado nível de processamento. Quando for identificado qual reestruturação se aplica melhor para determinado tipo de dado sísmico, é possível desenvolver compressores inteligentes que empreguem tais estratégias visando a reduzir ainda mais o tamanho do arquivo comprimidoado.

Em casos em que não se sabe em que estágio de processamento se encontra o dado sísmico, é possível utilizar estratégias de amostragem para selecionar uma parte pequena e representativa do dado sísmico e aplicar diversas estratégias de reestruturação. Desta forma, aquela que produzir a melhor taxa de compressão seria aplicada em todo o dado sísmico e, em seguida, ocorreria a compressão. É claro que este processo de seleção da estratégia é algo que pode elevar o tempo da compressão, porém poderia reduzir ainda mais o tamanho dos arquivos comprimidos, facilitando seu armazenamento e transmissão.

Para reduzir o tempo de compressão e descompressão, a melhor alternativa é recorrer ao paralelismo. Apesar dos algoritmos de compressão sem perdas serem difíceis de serem paralelizados, o compressor PBZip2 é uma prova de que é possível

reduzir significativamente o tempo de compressão e descompressão, pouco afetando as taxas de compressão. Além disso, a evolução tecnológica traz novos componentes, como a placa Intel Xeon Phi[1] que fornece um aumento significativo na capacidade de processamento. Desta forma, é possível criar mais unidades de processamento para reduzir cada vez mais o tempo de execução.

[1] `http://www.intel.com/content/www/us/en/processors/xeon/xeon-phi-detail.html`

Referências Bibliográficas

[1] BROWN, A. R., BROWN, A. R., BROWN, A. R., et al. "Interpretation of three-dimensional seismic data", 2004.

[2] YILMAZ, Ö. *Seismic Data Analysis*, v. 2. Tulsa, Society of Exploration Geophysicists, 2001.

[3] BACON, M., SIMM, R., REDSHAW, T. *3-D seismic interpretation*. Cambridge University Press, 2007.

[4] REYNOLDS, J. M. *An Introduction to Applied and Environmental Geophysics*. John Wiley & Sons, 2011.

[5] YILMAZ, Ö., DOHERTY, S. M. *Seismic Data Processing*, v. 2. Tulsa, Society of Exploration Geophysicists, 1987.

[6] SHERIFF, R. E., GELDART, L. P. *Exploration Seismology*. Cambridge University Press, 1995.

[7] DONOHO, P. L., VILLASENOR, J. D. "High-performance Seismic Trace Compression". out. 1995. Disponível em: <http://www.onepetro.org/mslib/servlet/onepetropreview?id=SEG-1995-0160>.

[8] HUFFMAN, D. A. "A Method for the Construction of Minimum-Redundancu Codes", *A Method for the Construction of Minimum-Redundancu Codes*, pp. 1098–1102, set. 1952.

[9] WITTEN, I. H., NEAL, R. M., CLEARY, J. G. "Arithmetic Coding For Data Compression", v. 30, pp. 520–540, jun. 1987.

[10] GENG, Y., WU, R.-S., GAO, J. "Dreamlet Transform Applied to Seismic Data Compression And Its Effects On Migration". Houston, Texas, out. 2009. Disponível em: <http://www.onepetro.org/mslib/app/Preview.do?paperNumber=SEG-2009-3640&societyCode=SEG>.

[11] NIKITIN, V. V., DUCHKOV, A. A., ANDERSSON, F. "Parallel algorithm of 3D wave-packet decomposition of seismic data: Implementation and optimization for GPU", *Journal of Computational Science*, v. 3, n. 6, pp. 469–473, nov. 2012. ISSN: 1877-7503. doi: 10.1016/j.jocs.2012.08.011. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877750312001007>.

[12] REDDY, T., DEVI, K., GANGASHETTY, S. "Nonlinear principal component analysis for seismic data compression". In: *2012 1st International Conference on Recent Advances in Information Technology (RAIT)*, pp. 927 –932, mar. 2012. doi: 10.1109/RAIT.2012.6194558.

[13] STEARNS, S., TAN, L.-Z., MAGOTRA, N. "Lossless compression of waveform data for efficient storage and transmission", *IEEE Transactions on Geoscience and Remote Sensing*, v. 31, n. 3, pp. 645 –654, maio 1993. ISSN: 0196-2892. doi: 10.1109/36.225531.

[14] STEARNS, S. D., KIRLIN, R. L., FAN, J. "Techniques for geophysical data compression", *SPIE 1941, Ground Sensing,*, v. 1941, pp. 212–220, ago. 1993. doi: 10.1117/12.154689.

[15] MANDYAM, G., MAGOTRA, N., MCCOY, W. "Lossless seismic data compression using adaptive linear prediction". In: *Geoscience and Remote Sensing Symposium, 1996. IGARSS '96. 'Remote Sensing for a Sustainable Future.', International*, v. 2, pp. 1029 –1031, Lincoln, NE, maio 1996. ISBN: 0-7803-3068-4. doi: 10.1109/IGARSS.1996.516556.

[16] STEARNS, S. "Arithmetic coding in lossless waveform compression", *IEEE Transactions on Signal Processing*, v. 43, n. 8, pp. 1874 –1879, 1995. ISSN: 1053-587X. doi: 10.1109/78.403346. 00058.

[17] STEARNS, S. "A Technique For Lossless Compression Of Seismic Data". In: *Geoscience and Remote Sensing Symposium, 1992. IGARSS '92. International*, v. 1, pp. 681 –683, 1992. doi: 10.1109/IGARSS.1992.576804.

[18] MCCOY, J., MAGOTRA, N., STEARNS, S. "Lossless predictive coding". In: *, Proceedings of the 37th Midwest Symposium on Circuits and Systems, 1994*, v. 2, pp. 927–930 vol.2, ago. 1994. doi: 10.1109/MWSCAS.1994.518963.

[19] HONEA, D., STEARNS, S. "Lossless waveform compression: a case study". In: *1993 Conference Record of The Twenty-Seventh Asilomar Conference on*

Signals, Systems and Computers, 1993, pp. 1514 –1518 vol.2, nov. 1993. doi: 10.1109/ACSSC.1993.342360.

[20] FAN, J., KIRLIN, R., STEARNS, S. "Enhancements of bi-level coding for seismic waveform compression". In: *, Proceedings of the 37th Midwest Symposium on Circuits and Systems, 1994*, v. 2, pp. 931 –934 vol.2, 1994. doi: 10.1109/MWSCAS.1994.518964.

[21] PETERSON, C., HUTT, C. "Lossless compression of seismic data". In: *1992 Conference Record of The Twenty-Sixth Asilomar Conference on Signals, Systems and Computers, 1992*, pp. 712 –716 vol.2, 1992. doi: 10.1109/ACSSC.1992.269103.

[22] NIJIM, Y., STEARNS, S., MIKHAEL, W. "Lossless compression of seismic signals using differentiation", *IEEE Transactions on Geoscience and Remote Sensing*, v. 34, n. 1, pp. 52 –56, jan. 1996. ISSN: 0196-2892. doi: 10.1109/36.481892.

[23] FARGUES, M., STEARNS, S., COUTU, G. "Seismic data compression using adaptive filters". In: *, Proceedings of the 37th Midwest Symposium on Circuits and Systems, 1994*, v. 2, pp. 957 –960 vol.2, 1994. doi: 10.1109/MWSCAS.1994.518970.

[24] NIJIM, Y., STEARNS, S., MIKHAEL, W. "Lossless compression of seismic signals using least square, frequency domain pole-zero modeling". In: *, 1995 IEEE International Symposium on Circuits and Systems, 1995. ISCAS '95*, v. 2, pp. 1106 –1109 vol.2, maio 1995. doi: 10.1109/ISCAS.1995.520340.

[25] NIJIM, Y., STEARNS, S., MIKHAEL, W. "Pole-zero modeling for the lossless compression of seismic signals". In: *, Proceedings of 1997 IEEE International Symposium on Circuits and Systems, 1997. ISCAS '97*, v. 4, pp. 2537 –2540 vol.4, jun. 1997. doi: 10.1109/ISCAS.1997.612841. 00000.

[26] ABANMI, A. O., ALSHEBEILI, S. A., ALAMRI, T. H. "Lossless compression of seismic data", *Journal of the Franklin Institute*, v. 343, n. 4–5, pp. 340–351, jul. 2006. ISSN: 0016-0032. doi: 10.1016/j.jfranklin.2006.02.016. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0016003206000391>.

[27] XIE, X., QIN, Q. "Fast Lossless Compression of Seismic Floating-Point Data". In: *International Forum on Information Technology and Applications,*

2009. IFITA '09, v. 1, pp. 235 –238, Chengdu, maio 2009. ISBN: 978-0-7695-3600-2. doi: 10.1109/IFITA.2009.556.

[28] SWELDENS, W. "The Lifting Scheme: A New Philosophy in Biorthogonal Wavelet Constructions". In: *in Wavelet Applications in Signal and Image Processing III*, pp. 68–79, 1995. 00000.

[29] WANG, X.-Z., TENG, Y.-T., GAO, M.-T., et al. "Seismic data compression based on integer wavelet transform", *Acta Seismologica Sinica*, v. 17, n. 1, pp. 123–128, nov. 2004. ISSN: 1000-9116, 1993-1344. doi: 10.1007/s11589-004-0075-4. Disponível em: <http://link.springer.com/article/10.1007/s11589-004-0075-4>.

[30] ZHENG, F., LIU, S. "A fast compression algorithm for seismic data from non-cable seismographs". In: *2012 World Congress on Information and Communication Technologies (WICT)*, pp. 1215 –1219, nov. 2012. doi: 10.1109/WICT.2012.6409260. 00000.

[31] MEFTAH, A., ANTONINI, M., BEN AMAR, C. "Lossless compression of 3D seismic data using a horizon displacement compensated 3D lifting scheme", 2010. doi: 10.1117/12.840186. Disponível em: <http://dx.doi.org/10.1117/12.840186>.

[32] AQRAWI, A. A. *Effects of Compression on Data Intensive Algorithms*. MSc thesis, Norwegian University of Science and Technology, Department of Computer and Information Science, Oslo, Norway, jun. 2010.

[33] AQRAWI, A., ELSTER, A. "Bandwidth Reduction through Multithreaded Compression of Seismic Images". In: *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)*, pp. 1730–1739, maio 2011. ISBN: 978-1-61284-425-1. doi: 10.1109/IPDPS.2011.330. 00008.

[34] AVERBUCH, A. Z., MEYER, F., STRÖMBERG, J.-O., et al. "Low Bit-Rate Efficient Compression for Seismic Data", *IEEE Transactions on Image Processing*, v. 10, n. 12, pp. 1801–1814, dez. 2001. ISSN: 1057-7149. doi: 10.1109/83.974565.

[35] ROSTEN, T., AMUNDSEN, L. "Seismic Data Compression And Its Effect On the Amplitudes". San Antonio, Texas, set. 2001. 00002.

[36] DONOHO, P. L., ERGAS, R. A. "Development of Seismic Data Compression Methods For Reliable, Low-noise, Performance". Houston, Texas,

nov. 1999. Disponível em: <http://www.onepetro.org/mslib/app/Preview.do?paperNumber=SEG-1999-1903&societyCode=SEG>.

[37] ERGAS, R. A., VILLASENOR, J. D., POLZER, R. S., et al. "Measuring Seismic Data Compression: What Losses Are Acceptable?" Denver, Colorado, nov. 1996. Disponível em: <http://www.onepetro.org/mslib/app/Preview.do?paperNumber=SEG-1996-2041&societyCode=SEG>.

[38] SAHA, S. "Image compression - from DCT to wavelets: a review", *Crossroads*, v. 6, n. 3, pp. 12–21, mar. 2000. ISSN: 1528-4972. doi: 10.1145/331624.331630. Disponível em: <http://doi.acm.org/10.1145/331624.331630>.

[39] VETTERLI, M., KOVACEVIC, J. *Wavelets and subband coding*, v. 87. Prentice Hall PTR Englewood Cliffs, New Jersey, 1995. 03646.

[40] AHMED, N., NATARAJAN, T., RAO, K. "Discrete Cosine Transform", *IEEE Transactions on Computers*, v. C-23, n. 1, pp. 90–93, jan. 1974. ISSN: 0018-9340. doi: 10.1109/T-C.1974.223784. 02764.

[41] SPANIAS, A., JONSSON, S., STEARNS, S. "Transform methods for seismic data compression", *IEEE Transactions on Geoscience and Remote Sensing*, v. 29, n. 3, pp. 407 –416, 1991. ISSN: 0196-2892. doi: 10.1109/36.79431.

[42] VERMEER, P., BRAGSTAD, H., ORR, C. "Aspects of Seismic Data Compression". Denver, Colorado, nov. 1996. Disponível em: <http://www.onepetro.org/mslib/app/Preview.do?paperNumber=SEG-1996-2031&societyCode=SEG>. 00008.

[43] BERNASCONI, G., VASSALLO, M. "Efficient data compression for seismic-while-drilling applications", *IEEE Transactions on Geoscience and Remote Sensing*, v. 41, n. 3, pp. 687 – 696, mar. 2003. ISSN: 0196-2892. doi: 10.1109/TGRS.2003.808896.

[44] MEYER, F. G. "Fast compression of seismic data with local trigonometric bases", pp. 648–658, 1999. doi: 10.1117/12.366820. Disponível em: <http://dx.doi.org/10.1117/12.366820>.

[45] WANG, Y., WU, R.-S. "Seismic data compression by an adaptive local cosine/sine transform and its effects on migration", *Geophysical Prospecting*, v. 48, n. 6, pp. 1009–1031, 1999. ISSN: 1365-2478. doi: 10.1046/j.1365-2478.2000.00224.x. Disponível em: <http://onlinelibrary.wiley.com/doi/10.1046/j.1365-2478.2000.00224.x/abstract>.

[46] WU, R.-S., WANG, Y. "New flexible segmentation technique in seismic data compression using local cosine transform", pp. 784–794, 1999. doi: 10.1117/12.366835. Disponível em: <http://dx.doi.org/10.1117/12.366835>.

[47] WANG, Y., WU, R.-S. "Improvements On Seismic Data Compression And Migration Using Compressed Data With the Flexible Segmentation Sc Hemefor Local Cosine Transform". Calgary, Alberta, ago. 2000. Disponível em: <http://www.onepetro.org/mslib/app/Preview.do?paperNumber=SEG-2000-2048&societyCode=SEG>.

[48] BOSMAN, C., REITER, E., CO, E. P. R. "Seismic Data Compression Using Wavelet Transforms". Washington, DC, set. 1993. Society of Exploration Geophysicists. Cited by 0019.

[49] REITER, E. C. "A Quantitative Comparison of 1,2 And 3 Dimensional Wavelet Compression Methods For Seismic Data". Society of Exploration Geophysicists, jan. 1996. Disponível em: <https://www.onepetro.org/conference-paper/SEG-1996-1630>.

[50] VASSILIOU, A., WICKERHAUSER, M. V. "Comparison of Wavelet Image Coding Schemes for Seismic Data Compression". In: *Wavelet Applications in Signal and Image Processing V*, San Diego, CA, out. 1997. SPI.

[51] KHENE, M., ABDUL-JAUWAD, S. "Adaptive seismic compression by wavelet shrinkage". In: *Proceedings of the Tenth IEEE Workshop on Statistical Signal and Array Processing, 2000*, pp. 544 –548, 2000. doi: 10.1109/SSAP.2000.870184.

[52] AL-MOOHIMEED, M. "Towards an efficient compression algorithm for seismic data". In: *Radio Science Conference, 2004. Proceedings. 2004 Asia-Pacific*, pp. 550 – 553, ago. 2004. doi: 10.1109/APRASC.2004.1422555.

[53] WU, W., YANG, Z., QIN, Q., et al. "Adaptive Seismic Data Compression Using Wavelet Packets". In: *IEEE International Conference on Geoscience and Remote Sensing Symposium, 2006. IGARSS 2006*, pp. 787 –789, ago. 2006. doi: 10.1109/IGARSS.2006.202. 00005.

[54] DUVAL, L. C., NGUYEN, T. Q. "Seismic data compression: a comparative study between GenLOT and wavelet compression". pp. 802–810, out. 1999. doi: 10.1117/12.366837. Disponível em: <http://spie.org/Publications/Proceedings/Paper/10.1117/12.366837>.

[55] DUVAL, L. C., OKSMAN, J., NGUYEN, T. Q. "A new class of filter banks for seismic data compression". In: *Annual International Meeting*, v. 18, pp. 1907–1910. SEG, Soc. Expl. Geophysicists, 1999. doi: http://dx.doi.org/10.1190/1.1820920. Disponível em: <http://dx.doi.org/10.1190/1.1820920>.

[56] DUVAL, L. C., NGUYEN, T. Q., TRAN, T. D. "On Progressive Seismic Data Compression using GenLOT". In: *Proc. Conf. Inform. Sciences Syst. (CISS)*, pp. 956–959, 1999. Disponível em: <http://thanglong.ece.jhu.edu/CISS/fa6.html>.

[57] DUVAL, L., BUI-TRAN, V., NGUYEN, T., et al. "GenLOT optimization techniques for seismic data compression". In: *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings*, v. 6, pp. 2111 – 2114 vol.4, 2000. doi: 10.1109/ICASSP.2000.859252.

[58] DUVAL, L. C., NAGAI, T. "Seismic data compression using GULLOTS". In: *Proc. International Conference on Acoustic Speech and Signal Processing (ICASSP)*, v. 3, pp. 1765–1768, 2001. doi: http://dx.doi.org/10.1109/ICASSP.2001.941282.

[59] ZIV, J., LEMPEL, A. "A universal algorithm for sequential data compression", *IEEE Transactions on Information Theory*, v. 23, n. 3, pp. 337–343, maio 1997. ISSN: 0018-9448. doi: 10.1109/TIT.1977.1055714. Disponível em: <http://dx.doi.org/10.1109/TIT.1977.1055714>.

[60] BURROWS, M., WHEELER, D. *A Block-sorting Lossless Data Compression Algorithm*. Relatório técnico, Systems Research Center, maio 1994.