COMPRESSÃO DE IMAGENS UTILIZANDO RECORRÊNCIA DE PADRÕES MULTIESCALAS COM SEGMENTAÇÃO FLEXÍVEL

Waldir Sabino da Silva Júnior

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

Prof. Eduardo Antônio Barros da Silva, Ph.D.

Prof. Gelson Vieira Mendonça, Ph.D.

Prof. Abraham Alcaim, Ph.D.

Prof. Weiler Alves Finamore, Ph.D.

Prof. Murilo Bresciani de Carvalho, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

DEZEMBRO DE 2004

SILVA JÚNIOR, WALDIR SABINO DA

Compressão de Imagens Utilizando Recorrência de Padrões Multiescalas com Segmentação Flexível [Rio de Janeiro] 2004

XIII,145 pp 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2004)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1.Compressão de Imagens

2.Casamento de Padrões Multiescalas

3.Quantização Vetorial

4.Otimização Taxa-distorção

5.União de Blocos

I.COPPE/UFRJ II.Título (série)

Agradecimentos

Primeiramente, agradeço à minha esposa Thyrsiana, pela sua paciência, compreensão, ajuda e pelo seu amor. Agradeço também a Deus, por permitir a chegada do nosso bebê.

Aos meus pais Sônia e Waldir, minhas Irmãs Grêta e Tâmisa, por ajudarem a desenvolver meus valores, meu caráter, pelo amor que têm por mim e por formarem a pessoa que sou hoje.

Agradeço às minhas primas Fabíola e Priscila, por todo o incentivo que me deram, aos meus tios Jorge Alfredo e Gildson, por me ajudarem no início da minha vida estudantil. Agradeço também a todos os meus queridos tios, tias, primos e primas.

Ao Professor Eduardo Barros (COPPE), pela sua paciência, disciplina e por suas sugestões e correções realizadas neste trabalho, e também aos Professores Gelson (COPPE) e Murilo (UFF). Aos Professores Cícero Costa, Marly Costa e Sandro Bitar, pelas suas orientações em iniciação científica e por me mostrarem o mundo da pesquisa.

Agradeço também à empresa Fucapi, por permitir a realização deste trabalho, e em especial a Niomar, Guajarino, Antônio L. e Frederico.

Ao amigo Eddie Lima, pelas suas explicações, sempre pertinentes e extremamente esclarecedoras.

Aos amigos Ricardo Cruz, pelos incentivos, conselhos e orientação acadêmica e Marco Rodrigues, por suas argumentações e curiosidades sobre o meu trabalho.

Aos amigos do Laboratório de Processamento de Sinais (LPS II): Tadeu, Baltar, Ana e Fábio, por tudo o que fizeram por mim no período em que estava no Rio de Janeiro, e aos amigos Glaiton e Rubem, por permitirem minha estadia em seu apartamento e por todo o incentivo que me deram.

Waldir Sabino da Silva Júnior

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

COMPRESSÃO DE IMAGENS UTILIZANDO RECORRÊNCIA DE PADRÕES MULTIESCALAS COM SEGMENTAÇÃO FLEXÍVEL

Waldir Sabino da Silva Júnior

Dezembro/2004

Orientadores: Eduardo Antônio Barros da Silva
Gelson Vieira Mendonça

Programa: Engenharia Elétrica

O método de compressão de sinais MMP (*Multi-dimensional Multiscale Parser*) é baseado na recorrência de padrões multiescala onde cada bloco do sinal de entrada é aproximado por um dicionário adaptativo sendo atualizado com versão dilatadas e contraídas de concatenações dos blocos previamente codificados sendo estruturado sob a forma de uma árvore de segmentação binária. Recentemente, se verificou que métodos estruturados com árvores binárias tem um desempenho superior quando, nesta estrutura, incorpora-se um cenário de união de seus nós folhas, chamado prune-join. Neste trabalho investigamos uma modificação no MMP que contempla tal estrutura, ou seja, estudamos o MMP em um cenário de união de blocos. Os resultados em relação ao MMP original, para imagens suaves, tiveram ganhos de 0,6dB de PSNR, e para imagens mistas os resultados se mantiveram.

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

COMPRESSION OF IMAGES BASED ON RECURRENT MULTISCALE PATTERNS WITH FLEXIBLE SEGMENTATION

Waldir Sabino da Silva Júnior

December/2004

Advisors: Eduardo Antônio Barros da Silva
Gelson Vieira Mendonça

Department: Electrical Engineering

The method of signal compression MMP (*Multi-dimensional Multiscale Parser*) is based on recurrency of multiscale patterns, where each block of the input signal is approximated by an adaptive dictionary, which is updated with expanded and compressed versions of concatenation of block previously codified. They are structured by a binary segmentation tree. Recently, it was verified that structured methods with binary trees have a better performance when a scenario of union of leave nodes, called prune-join, is incorporated in this structure. In this work, we investigate some changes in MMP in order to be consistent with such structure, i.e., we study MMP in a scenario with block union. Comparing to the original MMP, the results for smooth images reached improvement of 0.6dB in PSNR, and for mixed images the results remain almost the same.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Introdução

A humanidade, nas últimas décadas, tem gerado uma grande quantidade de informação na forma digital. Essa explosão ocorreu devido as desenvolvimento de tecnologias para seu armazenamento incluindo-se os *CDs*, *Hard Disc*, produção de CIs de alta densidade, e para a sua transmissão, onde inclui-se as fibras ópticas, TV digital, *cable modems*, etc. Mesmo diante de todo esse desenvolvimento essas tecnologias não suprem a necessidade humana pois esta cresce mais rapidamente. Sendo assim, torna-se necessária a manipulação desse volume de informação digital de tal forma a obtermos uma representação mais compacta. A ciência ou arte que realiza tal tarefa é chamada de *Compressão de Dados* e tais representações são obtidas através de estruturas peculiares existentes nos dados. O termo *dado* refere-se aos meios pelos quais a informações é conduzida, ou seja, dados podem ser caracteres em um arquivo de texto, sequência de números que são amostras de um sinal de voz, matrizes númericas que são os níveis de cinza de uma imagem, etc.

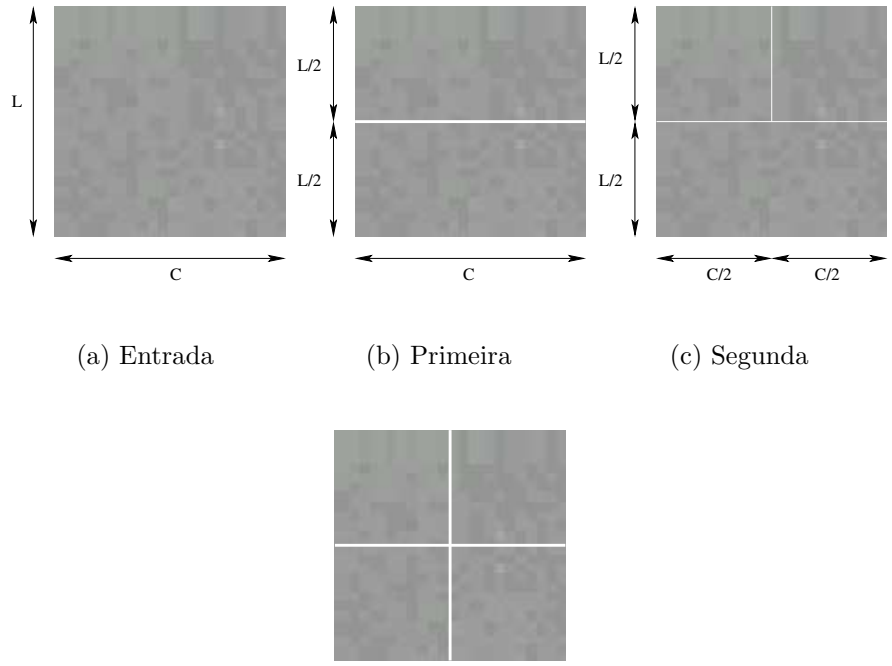
Em particular, quando estamos tratando de imagens é comum o uso do termo *Compressão de Imagens*. A pesquisa em compressão de imagens teve os primeiros trabalhos iniciados com o desenvolvimento de métodos analógicos para a compressão da largura de banda de transmissão de vídeo. Os métodos digitais começaram a partir do trabalho pioneiro de Shannon [1] que inseriu conceitos teóricos usados até então. O estado da arte em métodos de compressão de ima-

gem e vídeo baseiam-se em *Codificação por Transformada* [2, 3, 4]. Esses métodos são executados em três passos: o primeiro seria a aplicação de uma *transformada*, no segundo temos uma *quantização* e no terceiro a *codificação*. Atualmente, alguns padrões são amplamente difundidos, como por exemplo, o JPEG [5] que usa a transformada discreta do cosseno (DCT) [6], o JPEG2000 [7], o EZW [8] e o SPITH [9, 10] cuja transformada é a DWT (transformada wavelet) [11, 12, 13].

O sucesso desses métodos está intrinsecamente ligado à suposição de que uma imagem possui, essencialmente, a maioria de suas informações confinadas em baixas frequências. O método denominado MMP (*Multi-dimensional Multiscale Parser*) recentemente desenvolvido em [14, 15], dedica-se justamente à classes de imagens que não estão sobre esta suposição, como por exemplo, imagens misturadas com textos, chamadas imagens mistas.

O MMP é um método baseado na recorrência de padrões multiescalas, isto é, ele representa um bloco de dados de entrada usando versões dilatadas e contraídas de um dicionário. A medida que o processamento é efetuado o dicionário é atualizado somente com as concatenações de blocos previamente codificados.

Em particular, para um bloco 2D ($L \times C$), o MMP realiza partições horizontais e verticais e processa cada bloco separadamente. Podemos visualizar esse processamento através da figura 1.1. Assim podemos verificar a evolução para cada partição nas figuras 1.1(b), 1.1(c) e 1.1(d). De acordo com o crítério estabelecido a partição se encerra ou não. Em nosso caso a entrada será aproximada por quatro blocos do dicionário do MMP onde as linhas brancas definem seus limites. O arquivo compactado é obtido através da codificação dos índices dos blocos aproximados do dicionário e de *flags* de partição do bloco.



(a) Entrada (b) Primeira (c) Segunda

(d) MMP

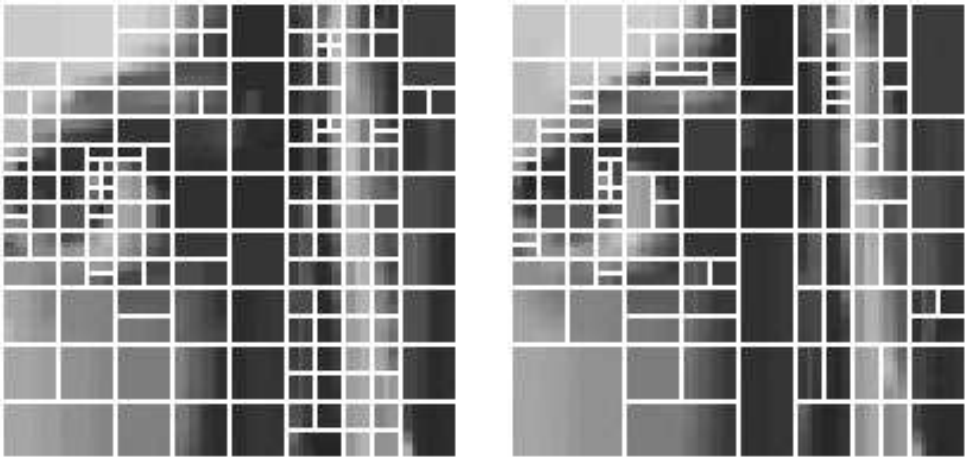
Figura 1.1: Evolução das partições até o bloco resultante

Considerando-se uma sub-imagem 64×64 (figura 1.2) da imagem *lena*, mostrada no apêndice B.1 desta dissertação. O MMP, após a realização do processamento dessa sub-imagem, obtém os blocos aproximados pelo dicionário com suas partições mostradas na figura 1.3(a). Assim, podemos perceber claramente a quantidade de blocos do dicionário usados para representar essa sub-imagem.

Figura 1.2: Bloco de entrada

Nossa proposta nessa dissertação é inserir o MMP em um *cenário de união* de blocos aproximados pelo dicionário e avaliar seu desempenho em relação ao MMP original. O cenário de união diz respeito a possibilidade de efetuarmos

uniões ou junções nos blocos aproximados pelo dicionário, tornando desta forma a partição mais geral. Podemos visualizar a união de blocos aplicada a sub-imagem *lena* através da figura 1.3(b). Além de diminuir o efeito de blocos temos, a cada união realizada, uma economia de um bloco a ser codificado.



(a) MMP Original (b) MMP União

Figura 1.3: Representação do bloco de entrada

1.2 Organização da dissertação

Esta dissertação está dividida em 7 capítulos da seguinte forma:

No Capítulo 2 apresentaremos um resumo teórico, onde serão abordados tópicos relacionados a compressão de sinais. Comentamos as técnicas de compressão de sinais existentes, assuntos relacionados à teoria da informação e as medidas consideradas para avaliação de desempenho.

No Capítulo 3 descrevemos o método de compressão de sinais MMP. Este capítulo contém uma introdução que comenta o funcionamento do MMP, a seguir temos o seu algoritmo, a otimização taxa-distorção e os resultados obtidos.

A seguir, no Capítulo 4 começamos a desenvolver nosso tema principal, onde apresentamos a primeira solução para o algoritmo de união, chamada de MMP-U1, iniciando com as principais motivações para sua criação, sua descrição, os resultados obtidos e a conclusão.

No Capítulo 5 realizamos a análise das estimativas de custos irreais disponibilizados pelo MMP-U1. Após isto propomos uma inversão no sentido de avaliação de união o qual soluciona o problema das estimativas.

A atualização do dicionário, para o cenário de união, é analisada no Capítulo 6. Neste verificamos sua ineficiência para um cenário de união se for realizada da mesma maneira que no MMP original. Diante disso investigamos uma maneira de tornar tal atualização mais eficiente, chamada de atualização reversa. Após isto, temos as conclusões e as referências bibliográficas.

Capítulo 2

Fundamentos de compressão de sinais

Neste capítulo trataremos de alguns fundamentos de compressão de sinais. Verificaremos quais são as técnicas existentes, alguns conceitos de teoria da informação e quais as medidas adotadas para avaliação de desempenho.

2.1 Técnicas de compressão de sinais

Quando usamos o termo *técnicas de compressão* (ou algoritmos de compressão) significa que estamos nos referenciando, basicamente, a um algoritmo de codificação, chamado de codificador, que recebe uma entrada X e produz uma saída X_c, e um algoritmo de decodificação, chamado decodificador, este recebe como entrada a saída produzida pelo codificador, ou seja, X_c, e gera uma saída Y. Tal esquema pode ser representado pela figura 2.1.

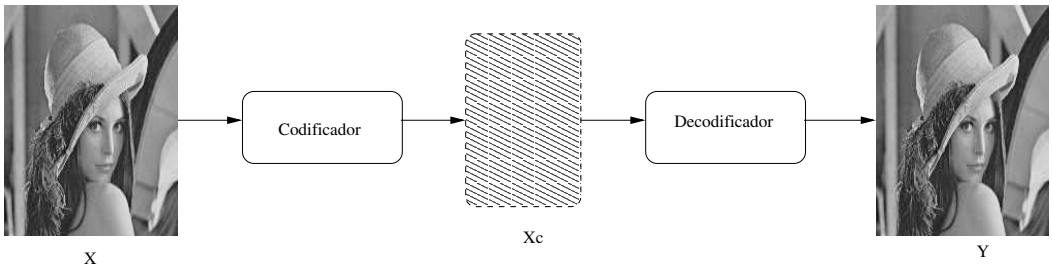


Figura 2.1: Esquema de Compressão

Baseando-se no resultado produzido pelo decodificador a técnica de compressão classifica-se em: *técnica de compressão de sinais sem perdas* e *com perdas*.

2.1.1 Compressão de sinais sem perdas

Neste tipo não existe diferença entre o resultado produzido pelo decodificador, ou seja Y, e a entrada do codificador, ou seja X. Podemos exemplificar tal técnica conforme a figura 2.2.

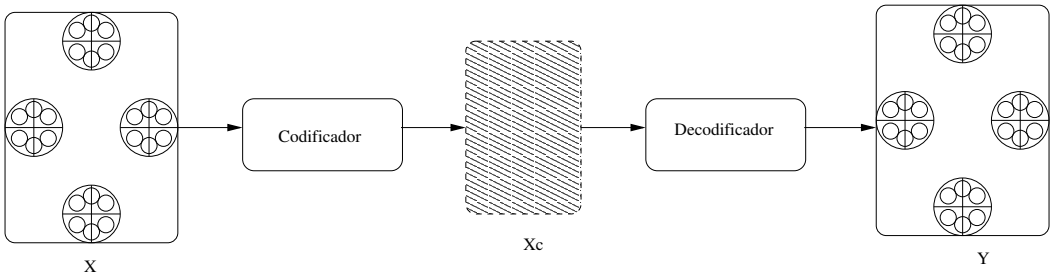


Figura 2.2: Compressão sem perdas

A compressão sem perdas não provoca distorcão na entrada X do codificador, ou seja, podemos recuperá-la completamente, sem perda alguma, quando efetuamos a decodificação. Pode-se dizer que a esta técnica não envolve perda de *informação* (o conceito de informação será visto na sessão 2.2). Uma característica bastante peculiar é que não se consegue uma taxa de compressão alta, ou seja, a relação entre o tamanho de X e Xc não possui um valor elevado.

Em muitos casos devemos usar uma compressão sem perdas; uma aplicação bastante comum é a compressão de informações que serão usadas para um diagnóstico médico, pois nessa situação, qualquer perda de informação pode implicar em um diagnóstico errado.

Existem técnicas bastante conhecidas para realizarmos uma compressão sem perdas, dentre as quais podemos destacar a técnica desenvolvida por David Huffman chamada de código de Huffman [16], o método chamado de codificação aritmética devido a Peter Elias descrito pela primeira vez em [17], e as técnicas baseadas em dicionários desenvolvidas por Abraham Lempel e Jacob Ziv [18, 19].

2.1.2 Compressão de sinais com perdas

Neste tipo, existe diferença entre o resultado produzido pelo decodificador, ou seja, Y, e a entrada do codificador, ou seja, X. Tal esquema é usado quando não existe a necessidade de integridade absoluta de reconstrução, ou seja, podemos aceitar algum tipo de perda de informação. Um exemplo pode ser visto na figura 2.3.

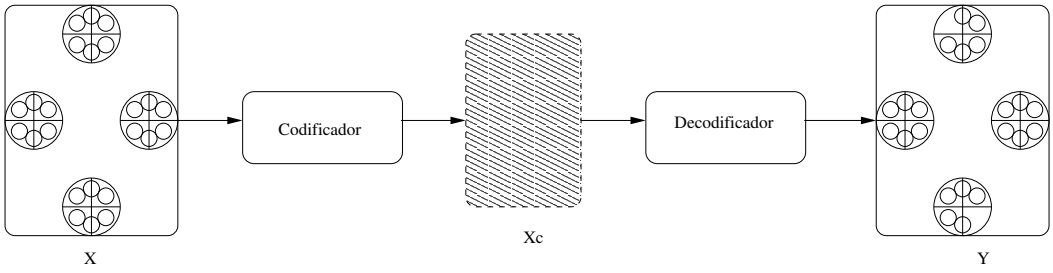


Figura 2.3: Compressão com perdas

Nos esquemas sem perdas temos, como principal medida, a taxa R (ver sessão 2.4), porém para esquemas com perdas, conforme mencionamos, existe algum tipo de perda de informação, portanto torna-se necessário outro tipo de medida para caracterizar essa perda ou diferença ocorrida entre X e Y, esta medida é chamada *distorção* D. A relação entre a taxa R e a distorção D é dada pela função taxa-distorção $R(D)$ descrita na sessão 2.3.

Os métodos de compressão com perdas são divididos em duas classes: *Quantização usando codificadores por entropia* e *Codificação por Transformada*, cada classe envolve um número limitado de *passos*, o primeiro método envolve duas partes e o segundo três.

Os métodos baseados em *Quantização usando codificadores por entropia* diminuem a faixa de valores assumidos pela fonte de entrada, para uma faixa muito pequena onde cada intervalo é representado por um código diferente e para cada código gerado associa-se um valor de reconstrução. Esses métodos são divididos em dois passos: a *quantização* e a *codificação*.

Os *Codificadores por Transformada* foram introduzidos por [2] e uma boa revisão encontra-se em [3, 4]. Nesses métodos efetua-se uma transformação da fonte de entrada de tal forma que o seu resultado contenha a maioria das informações

concentrada em poucos elementos, ou seja, efetuamos uma compactação da energia. Métodos baseados em *Codificação por Transformada* representam o estado da arte em compressão com perdas. Esses são divididos em três partes: a *transformação*, a *quantização* e a *codificação*.

A **transformação**, como o próprio nome já diz, consiste em transformarmos uma sequência de entrada $\{x_n\}$ em outra sequência $\{\theta_n\}$. Para recuperarmos a entrada, a partir do resultado da transformação, usamos a chamada transformação inversa. Ambas podem ser definidas, na forma matricial, como nas equações abaixo.

$$\theta = Ax \tag{2.1}$$

$$x = A^{-1}\theta \tag{2.2}$$

Definimos a transformação ortonormal para imagens e sua inversa como nas equações abaixo. Essas são ditas separáveis pois podemos transformar primeiramente suas linhas e após suas colunas.

$$\Theta = AXA^T \tag{2.3}$$

$$X = A^T\Theta A \tag{2.4}$$

Exemplos de transformadas são: A transformada wavelet (DWT) [11, 12, 13]; A transformada de walsh-hadamard (DWHT) [20], a transformada do cosseno (DCT) [6], dentre outras.

A **quantização** consiste no processo de representar a grande quantidade de símbolos fornecidos pela transformação usando uma quantidade muito menor de símbolos. Existem dois tipos de quantizadores: o *quantizador escalar* e o *vetorial*. A quantização escalar está caracterizada quando os valores de entrada são escalares; de outro modo, se os valores de entrada forem vetores então temos um quantizador vetorial. Podemos encontrar referências sobre quantizadores escalares em [21, 22] e sobre quantizadores vetoriais em [23, 24, 25].

A **codificação** é a última parte e consiste em codificar os símbolos, fornecidos pelo quantizador, com alguma técnida de codificação sem perdas, exemplos de tais técnicas são a codificação aritmética [17, 26, 27] e código de Huffman [16, 27].

■

Um esquema de compressão de sinais com perdas fornece uma compressão maior que um esquema sem perdas. Como iremos ver na sessão 2.2 um esquema sem perdas possui um limite teórico fundamental definido pela entropia da fonte. Para um esquema com perdas o que temos é uma relação entre a taxa e a distorção (ver sessão 2.3) portanto podemos conseguir uma taxa menor se aumentarmos a distorção.

2.2 Teoria da informação para compressão sem perdas

Conforme visto na sessão 2.1 as técnicas de compressão envolvem ou não perda de *informação*, pórem como podemos avaliar de forma quantitativa tal informação ? O primeiro estudo nesse sentido foi desenvolvido por Shannon em [1]. Neste, Shannon usou o termo *auto-informação* que pode ser definida da seguinte maneira:

Considerando-se um evento aleatório α e sua probabilidade associada $P(\alpha)$ então podemos definir a *auto-informação* associada a α, em bits, conforme a equação 2.5:

$$i(\alpha) = \log_2 \frac{1}{P(\alpha)} \tag{2.5}$$

Com esta definição pode-se concluir que: se a probabilidade de um evento é alta então a auto-informação é baixa, consequentemente, se a probabilidade de um evento é baixa então a auto-informação é alta.

Uma contribuição particularmente importante desenvolvida por Shannon foi a chamada *auto-informação média* mais conhecida como *entropia*, que pode ser definida conforme [28, 27, 29, 30, 31] como na equação 2.6 do seguinte modo:

Considerando-se uma fonte S com um alfabeto $A = \{a_1, a_2, \ldots, a_n\}$ onde a_i são os símbolos, com uma saída igual à $\{s_1, s_2, \ldots, s_m\}$ onde todos os $s_i \in A$ e suas probabilidades associadas são $P(s_i)$ então a *entropia*, em bits/símbolo, é definida como:

$$H(S) = \sum_{k=1}^{m} P(s_k) i(s_k)$$

Usando a equação 2.5 para substutuirmos $i(s_k)$ encontramos:

$$H(S) = -\sum_{k=1}^{m} P(s_k) \log_2 (s_k) \tag{2.6}$$

Conforme mostrado por Shannon esta quantidade estabelece um limite teórico para a compressão sem perdas. O melhor que pode-se realizar é codificar a saída da fonte S com um número médio de bits igual à entropia dessa fonte. Nas técnicas de compressão sem perdas a preocupação principal é obter códigos com taxas (ou seja, o número médio de bits usados para representar uma amostra, ver sessão 2.4) que se aproximam da entropia da fonte S.

2.3 Teoria da informação para compressão com perdas

Na sessão 2.2 vimos que a preocupação principal é a taxa de compressão e que o alfabeto produzido pela saída da fonte S é igual ao alfabeto de entrada. Quando consideramos a compressão com perdas temos duas diferenças básicas. A primeira delas é que a preocupação principal passa a ser a taxa e a *distorção*, tal *distorção*, expressa a diferença entre o resultado da decodificação (em nosso caso seria Y) e a entrada original (seria X). A taxa e a distorção são representadas por uma função de fundamental importância chamada *função taxa-distorção* geralmente escrita como $R(D)$ ou $D(R)$. A segunda é que os alfabetos de entrada e de saída são diferentes.

Para encontrarmos a função taxa-distorção $R(D)$ e também levarmos em consideração a diferença entre os alfabetos precisamos de dois conceitos adicionais, são eles: *entropia condicional* e *informação mútua média*.

Para definirmos a *entropia condicional* precisamos considerar duas variáveis aleatórias (ver [32]) X e Y, cujos valores, respectivamente, originam-se dos alfabetos $X' = \{x_1, x_2, \ldots, x_n\}$ e $Y' = \{y_1, y_2, \ldots, y_m\}$. Suas entropias, conforme a

equação 2.6 valem:

$$\begin{aligned} H(X) &= -\sum_{i=1}^{n} P(x_i)\log_2(x_i) \\ H(Y) &= -\sum_{j=1}^{m} P(y_j)\log_2(y_j) \end{aligned} \tag{2.7}$$

Assim, a entropia condicional é definida segundo [31] como:

$$H(X|Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i|y_j)P(y_j)\log_2 P(x_i|y_j) \tag{2.8}$$

Para definirmos a *informação mútua média* precisamos relembrar o conceito de *informação mútua* (ver [31]) que é dada, em bits, por:

$$i(x_k; y_l) = \log_2\left[\frac{P(x_k|y_l)}{P(x_k)}\right] \tag{2.9}$$

Assim a informação mútua média é, em bits/símbolo, igual a:

$$\begin{aligned} I(X;Y) &= \sum_{k=1}^{n}\sum_{l=1}^{m} P(x_k; y_l)i(x_k; y_l) \\ &= \sum_{k=1}^{n}\sum_{l=1}^{m} P(x_k; y_l)\log_2\left[\frac{P(x_k|y_l)}{P(x_k)}\right] \end{aligned}$$

Se usarmos a equação 2.7 encontramos:

$$I(X;Y) = H(X) - H(X|Y) \tag{2.10}$$

Desta forma, com as definições de entropia condicional (equação 2.8) e informação mútua média (equação 2.10) podemos seguir para encontrar a função taxa-distorção. O campo de Teoria da Informação que estuda tal assunto é conhecido como *teoria taxa-distorção*, nele o príncipal objeto de estudo é encontrar uma relação satisfatória entre a taxa R e a distorção D, isto, atualmente, é realizado pela *função taxa-distorção* $R(D)$. A função taxa-distorção especifica a menor taxa em que a saída da fonte pode ser codificada enquanto mantém a distorção menor que ou igual a D.

A definição matemática da função $R(D)$ encontra-se em [31] na forma de um teorema o qual será descrito abaixo.

Teorema 2.3.1 (Função taxa-distorção) *Dada uma fonte X identicamente distribuida com distribuição $P(x_i)$ e uma função distorção limitada $d(x_i, y_j)$ então a função taxa-distorção será dada por*

$$R(D) = \min_{P(y_j|x_i)\,|\,D \leq D^*} I(X;Y) \tag{2.11}$$

Onde a distorção média é $D = \sum_i \sum_j P(x_i)P(y_j|x_i)d(x_i, y_j)$ e D^* é a distorção alvo.

O comportamento da função taxa-distorção pode ser visto no gráfico 2.4. É importante observar que existem dois pontos bastante peculiares. O primeiro deles é o ponto definido por $(R_a, D_a) = (R_{max}, 0)$ neste temos distorção igual a zero, ou seja, estamos realizando uma codificação sem perdas. No segundo $(R_b, D_b) = (0, D_{max})$ temos a taxa igual a zero, ou seja, não estamos codificando informação alguma.

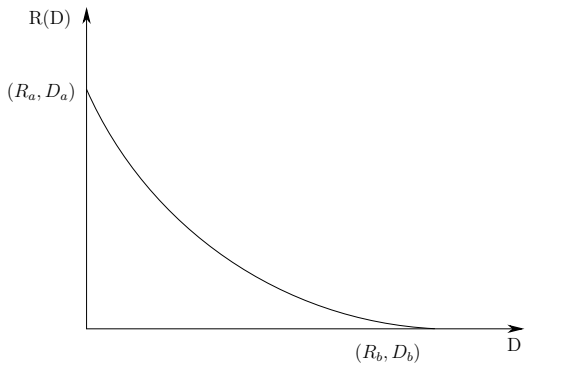


Figura 2.4: Função Taxa-Distorção $R(D)$

2.4 Medidas de desempenho

Nesta sessão vamos definir os critérios adotados para a realização dos testes de simulação. Todas as definições são voltadas para esquemas de compressão com perdas. Os sinais considerados são imagens em nível de cinza codificadas com 255 níveis distintos.

A **Taxa de compressão** é a nossa primeira medida, ela é definida como a relação entre o número de bits necessários para representar a imagem antes

da compressão (ou seja, seu tamanho original) e o número de bits necessários para representar a imagem depois da compressão (ou seja, seu tamanho depois da compressão). Tal relação pode ser visualizada pela equação 2.12.

$$CR = \frac{\text{Tamanho Original}}{\text{Tamanho Atual}} \tag{2.12}$$

A taxa de compressão pode ser medida em termos do número médio de bits necessários para representar um pixel, neste caso usa-se somente o termo **taxa** (R). A taxa pode ser obtida pela relação entre o produto da taxa atual pelo tamanho original da imagem e o seu tamanho atual. Podemos visualiza-la como:

$$R = \frac{[\text{Tamanho Original}] \cdot [\text{Taxa Atual}]}{\text{Tamanho Atual}} \tag{2.13}$$

Em nossa dissertação, a Taxa Atual é sempre igual a 8, pois as imagens de teste são codificadas a 8 bpp. Consequentemente, a taxa pode ser escrita conforme a equação 2.14:

$$R = \frac{[\text{Tamanho Original}] \cdot [8]}{\text{Tamanho Atual}} \tag{2.14}$$

■

A próxima medida é a **Distorção**. Em esquemas de compressão com perdas, conforme mencionamos, o sinal obtido na reconstrução não é igual ao original. Consequentemente, torna-se necessário uma medida para quantificar essa diferença, tal medida chama-se distorção. Nesta dissertação, nossa medida de distorção é a peak-signal-to-noise-ratio (PSNR). A PSNR é avaliada em dB e é definida em termos do erro médio quadrático σ^2 e do máximo nível de cinza da imagem considerada x_p. Sua equação é dada pela expressão 2.15.

$$\text{PSNR} = 10 \log_{10} \frac{x_p^2}{\sigma^2} \tag{2.15}$$

Supondo-se que a imagem $\{x_i\}$ e sua reconstrução $\{y_i\}$ possuem dimensões (N, M) então σ^2 vale (ver [27, 28, 29]):

$$\sigma^2 = \frac{1}{NM} \sum_{i=1}^{NM} (x_i - y_i)^2 \tag{2.16}$$

Logo podemos expressar a PSNR como:

$$\text{PSNR} = 10 \log_{10} \frac{x_p^2}{\frac{1}{NM} \sum_{i=1}^{NM} (x_i - y_i)^2} \tag{2.17}$$

onde, normalmente $x_p = 255$.

■

É importante notar que todos os críterios definidos são *críterios objetivos*. Tais critérios possuem a característica da associar valores para medir o nível de fidelidade da imagem original e a reconstruida. Sua principal vantagem é oferecer uma forma simples e conveniente para avaliação da perda de informação.

Pode-se usar critérios subjetivos para a avaliação, esses dependem inteiramente da observação humana e podem ser obtidos mostrando-se uma imagem e sua reconstrução para um grupo de observadores. Cada observador avalia o resultado da compressão de acordo com a escala definida em [33] e o resultado final é obtido com a média dessas avaliações.

Capítulo 3

Compressão de imagens usando recorrência de padrões multiescala (MMP)

Neste capítulo iremos descrever o algoritmo desenvolvido em [14] chamado de **MMP** (Multi-dimensional Multiscale Parser) que usa, como comentamos brevemente na introdução, casamento aproximado de padrões recorrentes. A seguir descreveremos uma introdução sobre os detalhes de funcionamento do MMP e um resumo do algoritmo em pseudo-código.

3.1 Introdução

O MMP é um método de compressão de sinais com perdas baseado na reccorrência de padrões multiescala. Nele temos a aproximação do bloco de entrada por elementos de um dicionário adaptativo seguindo um critério de controle, onde se tal críterio não é obedecido então o bloco é particionado e a tentativa de aproximação continua. O dicionário é atualizado à medida que o sinal está sendo processado, ou seja, temos um dicionário adaptativo. Outra caracteristica é que o critério de controle pode ser ajustado de modo a podermos efetuar uma compressão sem perdas. Em resumo podemos citas suas principais caracteristicas como:

- O MMP possui um dicionário adaptativo e sua inicialização não se baseia

em nenhum conhecimento prévio do sinal sendo assim, podemos dizer que o MMP possui um comportamento universal.

- O MMP pode ser ajustado para realizar uma compressão sem perdas.

O MMP fundamentalmente baseia-se em 3 pontos, são eles: *A inicialização do dicionário, o esquema de codificação e a atualização do dicionário* sendo vistos a seguir.

3.1.1 Inicialização do dicionário - parte 1

Nesta parte, estamos preocupados em descrever a estrutura do dicionário e a sua inicialização será vista somente na seção 3.1.5. O dicionário é composto por vários subdicionários de blocos S_k com dimensões específicas. O número de subdicionários é limitado e denifido em função da largura do bloco que será processado, ou seja, se considerarmos um bloco de entrada com dimensão 16×16 então as dimensões dos subdicionários serão dadas por $(16 \times 16), (8 \times 16), (8 \times 8), (4 \times 8), \ldots, (1 \times 1)$. Podemos visualizar o dicionário D do MMP e seus subdicionários S_k como na figura abaixo:

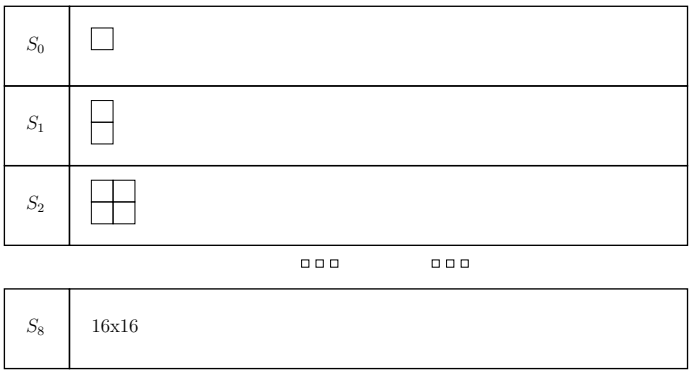


Figura 3.1: Dicionário D do MMP e seus subdicionários S_k

Para definirmos a sua inicialização é nessecário usarmos a transformação de escala que será definida somente na sessão 3.1.4, sendo assim, o algoritmo completo será visto somente nesta sessão.

3.1.2 Esquema de codificação

Quando usamos o MMP para efetuar a compressão de uma imagem temos uma codificação individual de cada bloco com dimensão $Lb \times Cb$, na Figura 3.2 vemos o primeiro bloco sendo codificado.

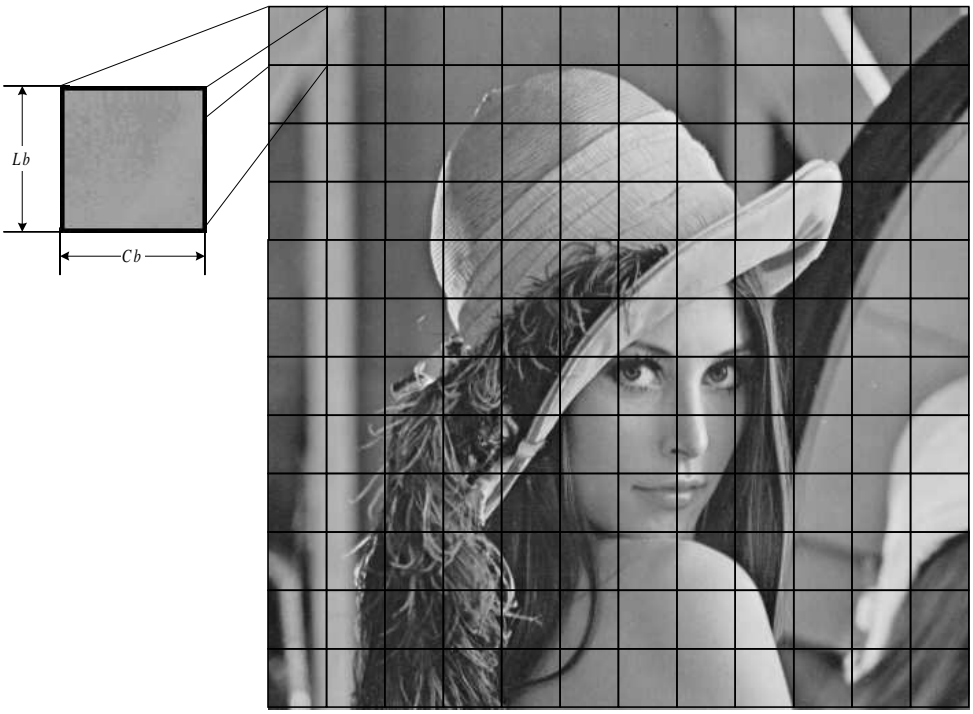


Figura 3.2: Esquema de Codificação do MMP

A codificação do bloco de dimensão $Lb \times Cb$ é realizada usando-se a melhor aproximação deste bloco pelo dicionário de mesma dimensão. A confirmação da codificação do bloco acontece quando esta aproximação satisfaz ao critério de controle, sendo assim codificamos um *flag* com valor 1 e o índice do bloco, que foi melhor aproximado do dicionário. Para visualizarmos este processo vamos considerar $(Lb, Cb) = 16 \times 16$. Por suposição, para este caso temos, como melhor aproximação, o bloco cujo índice é 63, como na Figura 3.3.

Quando o critério de controle não é satisfeito ocorre a partição horizontal do bloco, se o número de linhas for igual ao número de colunas ou a partição vertical caso contrário. Com isso geramos dois novos blocos com dimensão $\frac{Lb}{2} \times Cb$. Considerando-se nosso exemplo explicativo, teríamos dois novos blocos, ambos com dimensão $(Lb, Cb) = 8 \times 16$, sendo assim iríamos repetir o processo com os novos

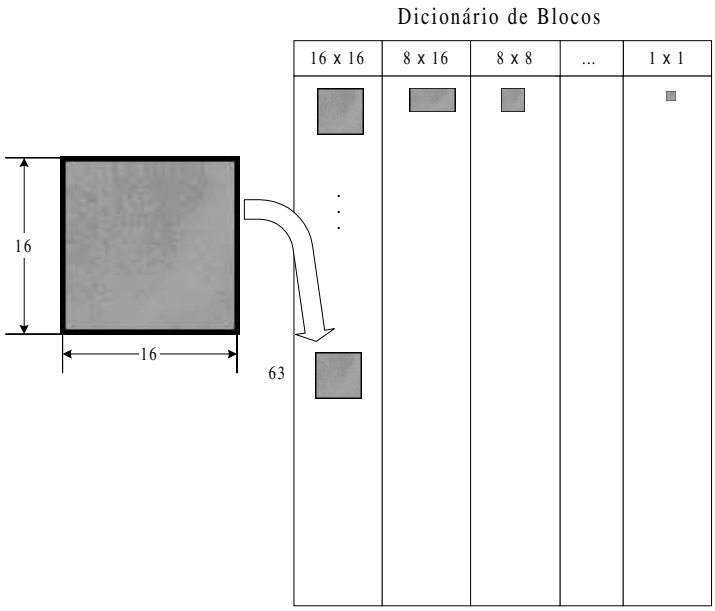


Figura 3.3: Achando melhor aproximação

blocos. Ver Figura 3.4.

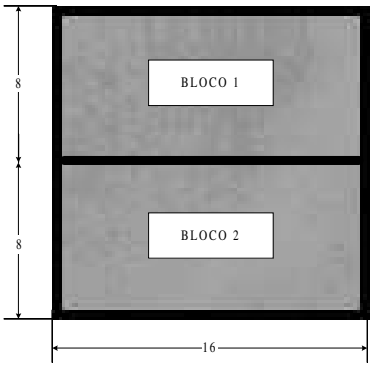


Figura 3.4: Partição horizontal

Devemos perceber que a procura pela melhor aproximação será realizada no dicionário de mesma dimensão, para nosso exemplo, esta dimensão é 8×16. A ordem dos blocos processados pelo MMP é sempre iniciada pelo bloco mais acima e mais a esquerda, portanto em nosso exemplo, o primeiro bloco processado pelo MMP será o BLOCO 1, como mostra a Figura 3.5.

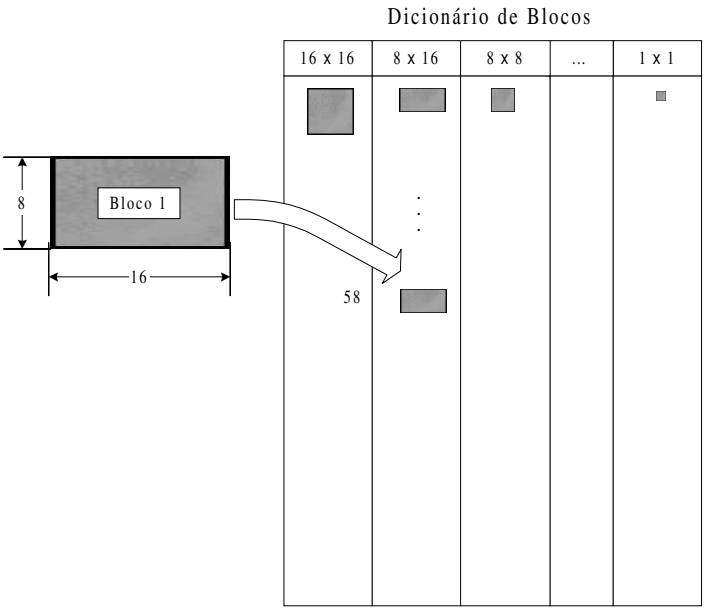


Figura 3.5: Aproximação quando existe a partição

3.1.3 Atualização do dicionário

A atualização do dicionário torna-se mais clara quando associamos as partições dos blocos processados pelo MMP a uma árvore binária. Cada bloco B^j de dimensão $(Lb, Cb) = N \times M$, quando particionado (critério de controle não satisfeito), gera dois blocos, B^{2j+1}, B^{2j+2} de dimensão $\left(\frac{Lb}{2}, Cb\right) = \frac{N}{2} \times M$ (se $N = M$), dessa forma, podemos associar o bloco B^j a um nó pai chamado n_j e os blocos particionados de, respectivamente, n_{2j+1} e n_{2j+2}. Assim geramos uma árvore à medida que os blocos são particionados. A seguir temos, nas Figuras 3.6, 3.7 e 3.8, a evolução das partições.

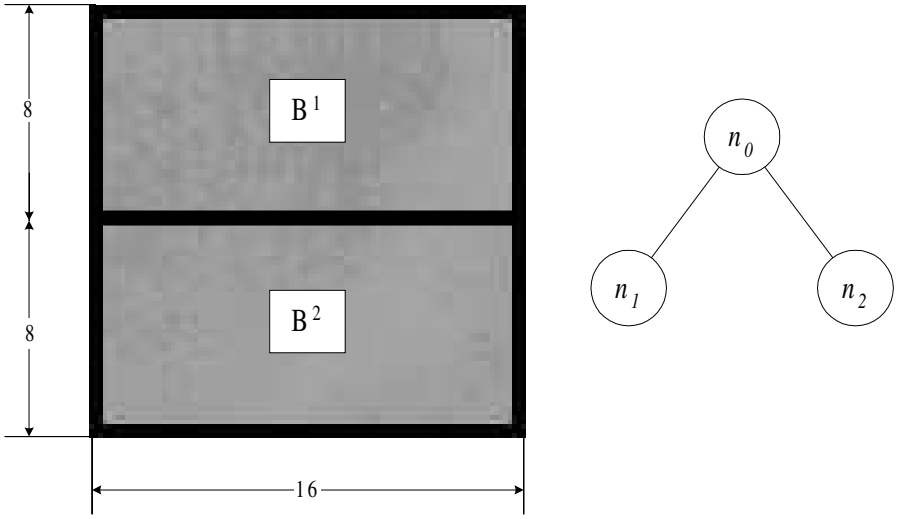


Figura 3.6: Analogia com árvore binária - primeira partição

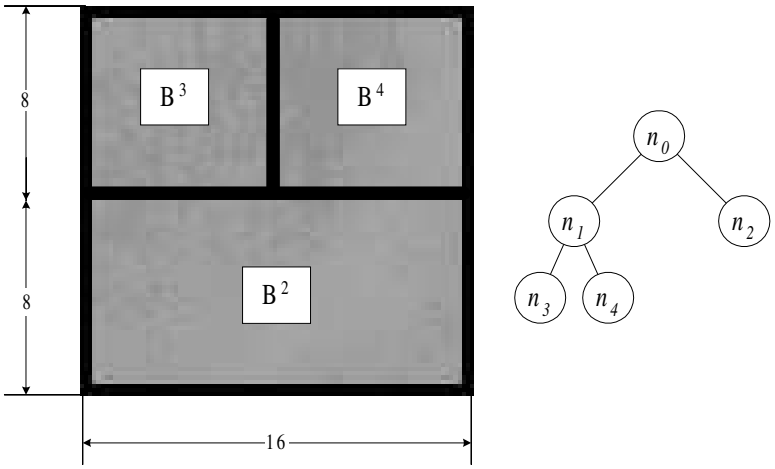


Figura 3.7: Analogia com árvore binária - segunda partição

Analisando-se desta forma podemos dizer que a atualização do dicionário é feita quando dois nós, filhos do mesmo nó pai, já estiverem sido codificados pelo MMP (chamados nós disponíveis). A atualização é realizada com a concatenação dos blocos codificados correspondentes aos nós disponíveis e com a *transformação* do bloco concatenado para todos os outros dicionários com dimensão diferente da

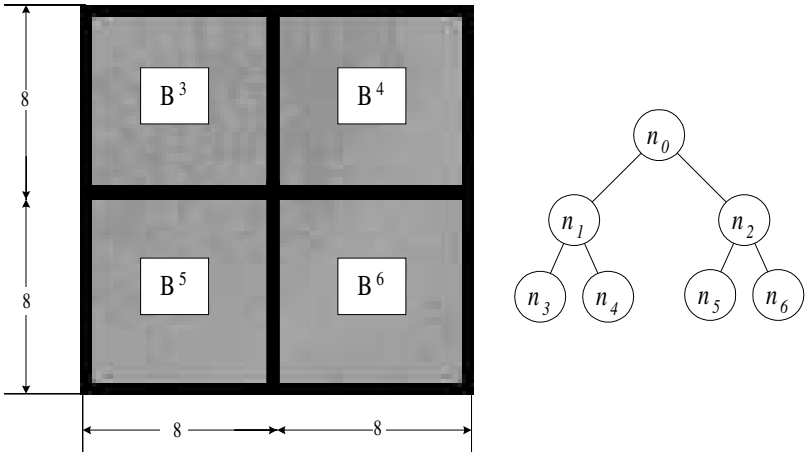


Figura 3.8: Analogia com árvore binária - terceira partição

dimensão do bloco concatenado.

Na Figura 3.9 temos a primeira atualização do dicionário. Como n_3 e n_4 já foram processados então ocorre a concatenação dos blocos correspondentes B^3 B^4 e suas transformações para os outros subdicionários. Nas figuras 3.10 e 3.11 temos as atualizações restantes.

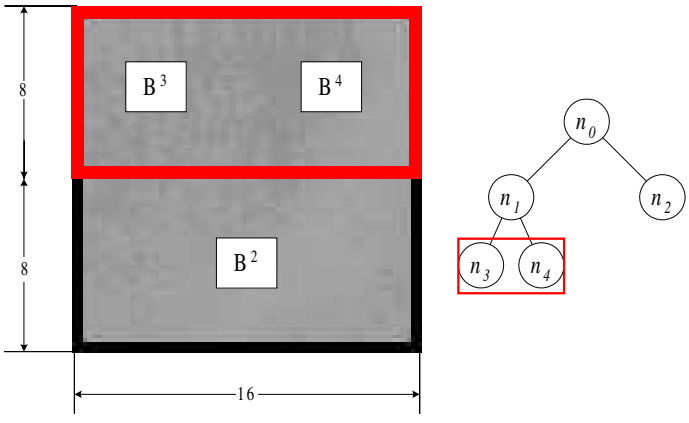


Figura 3.9: Primeira atualização do dicionário

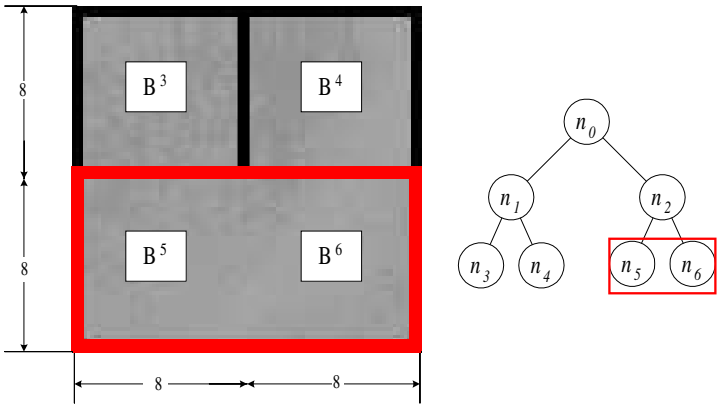


Figura 3.10: Segunda atualização

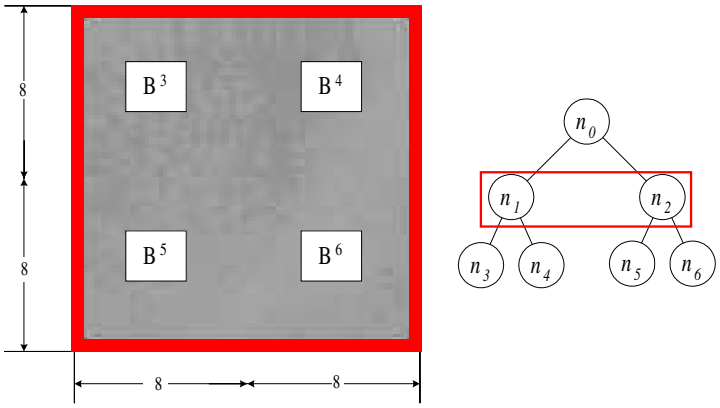


Figura 3.11: Terceira atualização

3.1.4 Transformação de Escala

Uma particularidade bastante importante do MMP é a *transformação das escalas* dos blocos usada para atualizar os dicionários com dimensões diferentes da dimensão do bloco concatenado. Basicamente esta transformação dilata ou contrai o bloco imagem. Conforme definido em [14], a transformação de escala de um bloco é definida, aplicando-se uma transformação unidimensional para todas as linhas do bloco em questão.

Uma transformação unidimensional é um operador que associa dois vetores de tamanhos distintos. Em particular, seja S um vetor de tamanho N_0 então podemos definir a transformação unidimensional de escalas como:

$$S^s = T_{N_0}^N [S] \tag{3.1}$$

onde N é o novo tamanho do vetor S.

Sendo assim, podemos dilatar/contrair vetores através da transformação de escalas, esta operação é realizada segundo [14] e [15] como descrito nas equações 3.2 e 3.3 abaixo.

- **Dilatação** ($N_0 < N$): Considerando-se $n = 0, 1, \ldots, N-1$. como cada coordenada do vetor transformado, podemos definir que o vetor transformado é dado por:

$$S_n^s = \left\lfloor \frac{\alpha_n \left(S_{m_n^1} - S_{m_n^0} \right)}{N} \right\rfloor + S_{m_n^0} \tag{3.2}$$

onde temos, com $\lfloor a \rfloor$ = menor inteiro maior que a:

$$\begin{aligned} m_n^0 &= \left\lfloor \frac{n\,(N_0 - 1)}{N} \right\rfloor \\ m_n^1 &= \begin{cases} m_n^0 + 1 & , \quad m_n^0 < N_0 - 1 \\ m_n^0 & , \quad m_n^0 = N_0 - 1 \end{cases} \\ \alpha_n &= n\,(N_0 - 1) - N m_n^0 \end{aligned}$$

- **Contração** ($N_0 > N$): Considerando-se $n = 0, 1, \ldots, N-1$. como cada coordenada do vetor transformado, podemos definir que o vetor transformado é dado por:

$$S_n^s = S_{m_{n,k=0}^0} + \frac{1}{N_0 + 1} \sum_{k=0}^{N_0} \left\lfloor \frac{\alpha_{n,k} \left(S_{m_{n,k}^1} - S_{m_{n,k}^0} \right)}{N} \right\rfloor \tag{3.3}$$

onde temos, com $\lfloor a \rfloor$ = menor inteiro maior que a:

$$\begin{aligned} m_{n,k}^0 &= \left\lfloor \frac{n\,(N_0 - 1) + k}{N} \right\rfloor \\ m_{n,k}^1 &= \begin{cases} m_{n,k}^0 + 1 & , \quad m_{n,k}^0 < N_0 - 1 \\ m_{n,k}^0 & , \quad m_{n,k}^0 = N_0 - 1 \end{cases} \\ \alpha_{n,k} &= n\,(N_0 - 1) + k - N m_{n,k}^0 \end{aligned}$$

Para o caso bidimensional, seja o bloco S de dimensão $N_0 \times M_0$ então definimos a transformação bidimensional em 3.4 como:

$$S^s = T^{N,M}_{N_0,M_0}[S] \tag{3.4}$$

onde $N \times M$ são as dimensões do novo bloco. A transformação bidimensional é implementada usando-se uma aplicação dupla do caso unidimensional onde, primeiramente, transformamos todas as linhas segundo a equação 3.5 com $l(S_i)$ sendo o tamanho de cada linha e o número de linhas variando de $i = 0, 1, \ldots, l(S_i) - 1$.

$$Y_i = T_M[S_i] \tag{3.5}$$

Realizamos a transformação das colunas usando a transposta de Y_i obtida na equação 3.5 com o número de colunas variando entre $j = 0, 1, \ldots, M - 1$.

$$S^s_j = \left(T_N\left[\left(Y^T\right)_j\right]\right)^T \tag{3.6}$$

3.1.5 Inicialização do Dicionário - parte 2

A inicialização do dicionário é realizada de acordo com o algoritmo abaixo. A quantidade de blocos, para cada subdicionário é dada por $NUMSEQI$ e este valor é inversamente proporcional a distorção d^*, a diferença entre os blocos é dada pela variável *passo*. No **passo 2** temos o uso da tranformação de escala definida na seção anterior. O número w de subdicionários é dado por $w = 2\log_2(M_{max}) + 1$, onde M_{max} é o valor máximo entre as dimensões dos subdicionários.

Passo 1: Inicialize as seguintes variáveis

* $NUMSEQ = \lfloor 2\frac{\max(I) - \min(I)}{\sqrt{d^*}+1} \rfloor$
* $passo = \frac{\max(I) - \min(I)}{NUMSEQ - 1}$

Passo 2: Inicialize o dicionário D da seguinte forma:

Para $n = (0, 1, \ldots, w - 1)$ **faça**

$N_x = 2^{\lfloor 0.5(n+1) \rfloor}$

$M_x = 2^{\lfloor 0.5n \rfloor}$

$S_{N_x,\,M_x} = \{S_{N_x,\,M_x}\} \cup \{T^{1,1}_{N_x,M_x}[n\,passo]\}$

3.2 Algoritmo MMP

Conforme mencionamos, um esquema de compressão é composto pelo codificador e o decodificador. O algoritmo de codificação MMP, de uma forma geral, tenta representar o bloco de entrada por algum padrão do dicionário de mesma dimensão sendo que essa tentativa segue o critério de distorção. Se tal critério é satisfeito então temos um nó folha da árvore de segmentação, caso contrário o bloco é particionado em dois outros e a tentativa continua, assim como a árvore de segmentação. O dicionário é atualizado com as concatenações das aproximações retornadas.

O decodificador lê a árvore de segmentação transmitida e os índices dos blocos do dicionário com isso reconstruimos a imagem. A atualização do dicionário é realizada da mesma forma.

A seguir iremos verificar o algoritmo de codificação para o MMP (ou seja o codificador) e o de decodificação originalmente apresentado em [14]. Podemos encontrar outras referências em [34, 15]. Considerando-se que:

1. O bloco original B^j, possui uma dimensão $(L_b, J_b) = (N, M)$ e j representa o nó n_j da árvore de segmentação $A(n_0)$ associada ao bloco codificado $\overline{B^j}$.

2. Um Dicionário $D = \{S_0, S_1, \ldots, S_w\}$ de blocos, inicializados da forma descrita na sessão 3.1 tanto para o codificador quanto para o decodificador. Cada subdicionário S_k pode ser acessado através da dimensão (N, M) do bloco B^j considerado. O número w de subdicionários é dado por $w = 2\log_2(M_{max}) + 1$.

3. Uma transformação de escala definida, para B^j (N, M) da forma abaixo conforme a equação 3.4 onde (N_x, M_x) são as novas dimensões.

$$T_{N,M}^{N_x,M_x}\left[B^j\right] \tag{3.7}$$

4. Um valor do critério de controle, chamado de distorção alvo d^*.

Procedimento $\overline{B^j}$ = `Codifica`(N, M, B^j, d^*)

Passo 1: Localizar o índice i do bloco s_i dentro do subdicionário S_x com dimensão (N, M) (mesma de B^j) que melhor aproxima B^j segundo o MSE dado por $\frac{1}{NM}(B^j - s_i)^2$. Guarde este valor em *erro* e faça $\overline{B^j} = s_i$.

Passo 2: **Se**($N = M = 1$) **então** guarde o índice i e retorne o bloco aproximado $\overline{B^j}$.

Se Não Siga para o passo 3.

Passo 3: **Se**(o critério de distorção é satisfeito, isto é $erro \leq d^*$) **então** guarde um *flag* igual a 1, o índice i e retorne o bloco $\overline{B^j}$.

Se Não Siga para o passo 4.

Passo 4: Grave um *flag* igual a 0.

Se($M > N$) **entao** divida B^j, em dois blocos, B^{2j+1} e B^{2j+2}, mudando suas dimensões para $\frac{M}{2}$ e chame o procedimento de codificação na ordem:

$\overline{B^{2j+1}}$ = `Codifica`(N, $\frac{M}{2}$, B^{2j+1}, d^*)

$\overline{B^{2j+2}}$ = `Codifica`(N, $\frac{M}{2}$, B^{2j+2}, d^*)

Se Não divida B^j, em dois blocos, B^{2j+1} e B^{2j+2}, mudando suas dimensões para $\frac{N}{2}$ e chame o procedimento de codificação na ordem:

$\overline{B^{2j+1}}$ = `Codifica`($\frac{N}{2}$, M, B^{2j+1}, d^*)

$\overline{B^{2j+2}}$ = `Codifica`($\frac{N}{2}$, M, B^{2j+2}, d^*)

Passo 5: Realize a concatenação dos blocos $\overline{B^{2j+1}}$ e $\overline{B^{2j+2}}$ da seguinte forma:

Se($M > N$) **entao** concatene na vertical, ou seja:

$$\overline{B^j} = \begin{pmatrix} \overline{B^{2j+1}} \\ \overline{B^{2j+2}} \end{pmatrix}$$

Se Não concatene na horizontal, ou seja:

$$\overline{B^j} = \begin{pmatrix} \overline{B^{2j+1}} & \overline{B^{2j+2}} \end{pmatrix}$$

Passo 6: Atualize o dicionário D da seguinte forma:

Para $n = (0, 1, \ldots, w-1)$ **faça**

$$N_x = 2^{\lfloor 0.5(n+1) \rfloor}$$
$$M_x = 2^{\lfloor 0.5n \rfloor}$$
$$S_{N_x, M_x} = \{S_{N_x, M_x}\} \cup \left\{ T^{N,M}_{N_x, M_x} \left[\overline{\overline{B^j}} \right] \right\}$$

Onde: $\lfloor a \rfloor$ representa o maior inteiro que é menor ou igual a a.

Procedimento $\overline{B^j}$ = `Decodifica`(N, M)

Passo 1: **Se** ($N = M = 1$) **então** leia um índice i. Acesse o dicionário S_x de dimensão (1,1), faça $\overline{B_j} = s_i$ e retorne $\overline{B_j}$.

Passo 2: Leia um *flag* e armazene em *varflag*.

Passo 3: **Se** (*varflag* = 1) **então** leia um índice i. Acesse o dicionário de dimensão (N, M) faça $\overline{B_j} = S_i$ e retorne $\overline{B_j}$.

Se não Siga para o passo 4.

Passo 4: **Se**($M > N$) **entao** faça

$\overline{B^{2j+1}}$ = `Decodifica`(N, $\frac{M}{2}$)

$\overline{B^{2j+2}}$ = `Decodifica`(N, $\frac{M}{2}$)

Se Não faça

$\overline{B^{2j+1}}$ = `Decodifica`($\frac{N}{2}$, M)

$\overline{B^{2j+2}}$ = `Decodifica`($\frac{N}{2}$, M)

Passo 5: Idêntico ao **Passo 5** do procedimento `Codifica`.

Passo 6: Idêntico ao **Passo 6** do procedimento `Codifica`.

3.3 MMP com Otimização Taxa-Distorção

3.3.1 Introdução

Conforme mencionamos, o MMP usa um critério de distorção para decidir se um bloco será codificado ou não. A título de exemplo podemos considerar a seguinte árvore $A(n_0)$ cuja distorção vale $d^* = 29$. Cada nó possui a indicação de menor distorção encontrada e os *flags* e índices que serão codificados. Veja a figura 3.12.

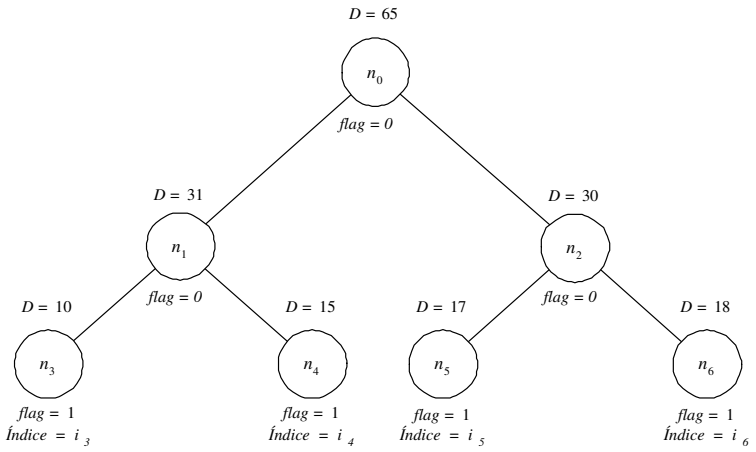
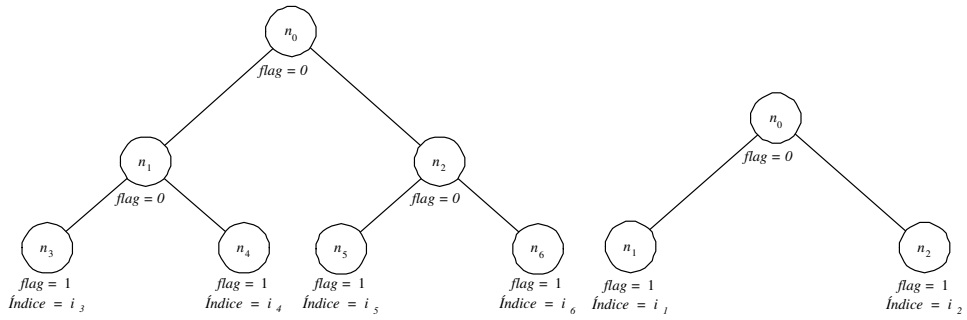


Figura 3.12: Árvore e suas distorções

Para nossa árvore acima temos, claramente que os nós n_1 e n_2 possuem uma distorção bem próxima da desejada, porém como esta ainda é superior então decide-se pela partição desses nós. O impacto dessa escolha é imediato pois ao invés de codificarmos 03 *flags* e 02 índices, ver figura 3.13(a), teremos que codificar 07 *flags* e 04 índices, ver figura 3.13(b). Assim podemos perceber que tal critério é restrito apenas ao bloco em questão, ou seja, o critério de decisão é local e não leva em consideração o consumo de *flags* e índices utilizados para codificar tal bloco.

Portanto surge a necessidade de avaliar a *distorção* $\times$ *consumo de bits necesários para codificação*, ou seja, *distorção* $\times$ *taxa*, não esquecendo que tal critério não deve ser restrito e sim estendido para toda a árvore $A(n_0)$, ou seja, um critério global. Essa avaliação é feita por uma variação do MMP, o chamado MMP-RDI, que será descrito a seguir.



(a) Critério Local (b) Economia

Figura 3.13: Problema do Critério Local

3.3.2 Descrição do MMP-RDI

Existem dois modos de avaliarmos a *distorção* × *taxa*. O primeiro deles surgiu do trabalho original realizado por [14], chamado de MMP-RD e o segundo por [34], chamado de MMP-RDI. Em ambos temos um critério de decisão global e desempenhos equivalentes porém a complexidade computacional do MMP-RDI é menor quando comparada ao MMP-RD.

O critério de decisão global é baseado no *custo lagrangeano* [35]. A otimização pelo método de lagrange foi primeiramente introduzida por [36], podemos encontrar aplicações desse método em [37, 38, 39]. O custo lagrangeano define uma relação entre a distorção e a taxa para um nó em particular sendo definido por:

$$J_{n_x} = D_{n_x} + \lambda RI_{n_x} \tag{3.8}$$

Onde:

- J_{n_x}: é o custo para o nó n_x.
- D_{n_x}: é a distorção entre o bloco B_x associado ao nó n_x e a sua aproximação s_i do subdicionário S com a mesma dimensão de B_x, onde i é o índice de S. A expressão da distorção é dada por:

$$D_{n_x} = \sum_{i=1}^{NM} (B_x - s_i)^2$$

- RI_{n_x}: é a taxa para representar o índice i. Sua expressão é dada por $RI_{n_x} = -\log_2 P(n_x)$ com $P(n_x)$ sendo igual a probabilidade de ocorrência de n_x. Logo pode ser calculado como:

$$RI_{n_x} = \log_2 \left(\frac{\text{total acumulado para } i}{\text{freqência de } i} \right)$$

- λ: é o fator ponderador entre a taxa e distorção.

Esse critério permite a *poda* ou *descarte* dos nós de acordo com o seu custo, podemos dizer que a árvore de segmentação é moldada de acordo com a seguinte premissa: *Um nó n_x somente será particionado em n_{2x+1} e n_{2x+2} se o custo diminuir.*

A principal característica do MMP-RDI é realizar todos os passos executados na codificação, ou em outras palavras podemos dizer que estamos fazendo um espelho. O comportamento espelhado é executado pelo procedimento `OtimizacaoRDI` e somente após determinarmos qual o melhor molde para a árvore considerada é que efetuamos a codificação, sendo realizada pelo procedimento `Codifica` com algumas diferenças, a decodificação não sofre nenhuma alteração e pode ser realizada pelo procedimento `Decodifica` e a inicialição do dicionário sofreu uma pequena alteração. A seguir apresentaremos algumas considerações, os procedimentos `OtimizacaoRDI` e `Codifica` e a inicialização do dicionário.

- O dicionário D é simulado através do dicionário rascunho D_R onde todas as atualizações serão simuladas e armazenadas nem D_R.

- É necessário um controle bastante estável da contagem das freqüências dos índices e dos *flags* de segmentação iguais a 0 e 1 para um funcionamento adequado do MMP-RDI. Abaixo temos o modelo usado para a realização deste controle.

Modelo para os Índices:

Tabela 3.1: Modelo Estatístico para o índice do dicionário

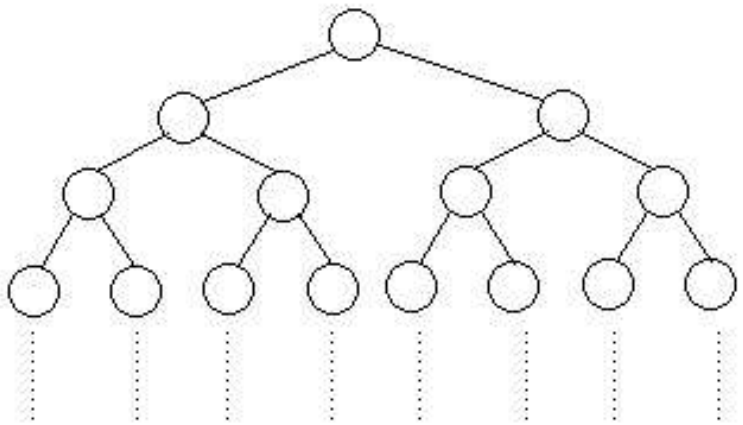
Sigla	Comentários
fi	Modelo de frequências para o dicionário D. No Procedimento `OtimizacaoRDI` seus valores, em nenhum momento, são alterados.
fio	Frequência espelho dos índices do dicionarios D. Esta complementa fi. Este contador simula o comportamento da contagem para os índices dentro da `OtimizacaoRDI`.
fir	Modelo de frequências para os índices do dicionário rascunho D_R. Este é usado para contar os índices dos blocos que surgiram das simulações das atualizações.

Modelo para os Flags de segmentação:

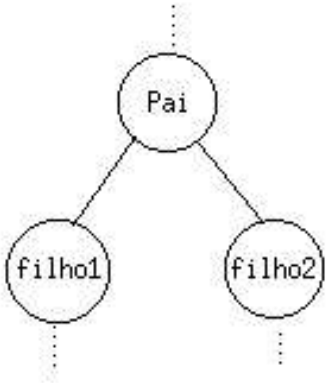
Tabela 3.2: Modelo Estatístico para os *flags* de segmentação

Sigla	Comentários
f_fs	Modelo de frequência para os *flags* de segmentação (0 e 1). Não são usados na `OtimizacaoRDI`.
f_fs_o	Complementa a frequência dos *flags* de segmentação. Este contador simula o comportamento da contagem para os *flags* dentro da `OtimizacaoRDI`.

- O MMP-RDI inicia com uma árvore $A(n_0)$ com todos os seus possíveis nós folhas, chamada de $\overline{A(n_0)}$ (veja a figura 3.14(a)). A partir dos nós folhas a árvore é moldada seguindo-se a premissa descrita acima. De uma maneira genérica esse molde pode ser visualizado da como na figura 3.14(b), ou seja, se o custo do *pai* é menor que ou igual a soma dos custos dos *filhos* então a sub-árvore é descartada e o nó *pai* passa a ser o uma folha.



(a) Árvore cheia - $\overline{A(n_0)}$



(b) Sub-árvore

Figura 3.14: Árvore para o MMP-RDI

Procedimento $\left[\overline{B^j},\, A(n_0)\right]$ = `OtimizacaoRDI`(N, M, B^j, no, $\overline{A(n_0)}$)

Passo 1: Faz $A(n_0) = \overline{A(n_0)}$.

Passo 2: Localizar o índice i do bloco s_i dentro dos subdicionários $S_x \in D$ com dimensão (N, M) (mesma de B^j) que melhor aproxima B^j segundo o custo $J_{n_j} = D_{n_j} + \lambda RI_{n_j}$. Guarde i e faça $\overline{B^j} = s_i$. O custo é dado por:

$$J_{n_j} = \left(s_i - B^j\right)^2 + \lambda \log_2 \frac{\sum fi + \sum fio + \sum fir}{fi + fio}$$

Passo 3: Verificar se algum bloco s_{i_R} dentro dos subdicionários $S_{Rx} \in D_R$ possuem uma aproximação melhor, em relação ao custo, que a escolhida no **Passo 2**. Se isso acontecer, guarde i e faça $\overline{B^j} = s_{i_R}$. O custo é dado por:

$$J_{n_j} = \left(s_i - B^j\right)^2 + \lambda \log_2 \frac{\sum fi + \sum fio + \sum fir}{fir}$$

Passo 4: **Se** (N = M = 1) **então**

Se (A aproximação foi feita por D) **então** Incrementa fio.

Se (A aproximação foi feita por D_R) **então** Incrementa fir.

Retorne $\overline{B^j}$ e $A(n_0)$.

Se não siga para o **Passo 5**.

Passo 5: Acrescenta, ao custo J_{n_j}, o valor da taxa para o *flag* de segmentação igual a 1, λR_1, conforme a equação abaixo. Esse valor representa completamente o custo para um nó folha.

$$\lambda R_1 = \lambda \log_2 \frac{\sum f\_fs(1) + \sum f\_fs\_o(1)}{f\_fs(1) + f\_fs\_o(1)}$$

Passo 6: Calcule e armazene, o valor da taxa para o *flag* de segmentação igual a 0, λR_0, conforme a equação abaixo. Armazene este valor no vetor de nome $\lambda R_0(no)$.

$$\lambda R_0 = \lambda \log_2 \frac{\sum f\_fs(0) + \sum f\_fs\_o(0)}{f\_fs(0) + f\_fs\_o(0)}$$

Passo 7: Incremente o *flag* de segmentação 0, ou seja, $f\_fs\_o(0)$.

Passo 8: **Se**$(M > N)$ **entao** divida B^j, em dois blocos, B^{2j+1} e B^{2j+2}, mudando suas dimensões para $\frac{M}{2}$ e chame a `OtimizacaoRDI` na ordem:

$\left[\overline{B^{2j+1}},\, A(n_0)_a\right] =$ `OtimizacaoRDI(` $N,\, \frac{M}{2},\, B^{2j+1},\, 2\,no+1,\, A(n_0)$)

$\left[\overline{B^{2j+2}},\, A(n_0)_b\right] =$ `OtimizacaoRDI(` $N,\, \frac{M}{2},\, B^{2j+2},\, 2\,no+2,\, A(n_0)$)

Se Não divida B^j, em dois blocos, B^{2j+1} e B^{2j+2}, mudando suas dimensões para $\frac{N}{2}$ e chame o procedimento de `OtimizacaoRDI` na ordem:

$\left[\overline{B^{2j+1}},\, A(n_0)_a\right] =$ `OtimizacaoRDI(` $\frac{N}{2},\, M,\, B^{2j+1},\, 2\,no+1,\, A(n_0)$)

$\left[\overline{B^{2j+2}},\, A(n_0)_b\right] =$ `OtimizacaoRDI(` $\frac{N}{2},\, M,\, B^{2j+2},\, 2\,no+2,\, A(n_0)$)

Passo 9: Faz $A(n_0) = A(n_0)_a$ **e** $A(n_0)_b$.

Passo 10: Se o custo do nó pai for menor ou igual ao custo dos nós filhos então não devemos continuar a segmentação. Matematicamente temos:

$$\{J_{no}\} \leq \left\{J_{(2\,no+1)} + J_{(2\,no+2)} + \lambda R_0(no)\right\}$$

Se a sentença for verdadeira siga para o **Passo 11**.

Se não vá para o **Passo 15**.

Passo 11: Faça Decrementos:

* $f\_fs\_o(0)$: decremento do *flag* 0 para as dimensões relacionadas as árvores $A(n_{2\,no+1})$ e $A(n_{2\,no+2})$ e para a dimensão N,M atual.
* $f\_fs\_o(1)$: decremento do *flag* 1 para todas as dimensões relacionadas as árvores $A(n_{2\,no+1})$ e $A(n_{2\,no+2})$.
* Decremento dos índices usados pelos nós que serão podados, se eles $\in D$ então decremente fio, e se $\in D_R$ então decremente fir.

Passo 12: Faça os incrementos:

* $f\_fs\_o(1)$: incrementa o *flag* 1 para o nível atual.
* Incrementa o índice usado pelo nó da dimensão atual se ele $\in D$ então incremente fio, e se $\in D_R$ então decremente fir.

Passo 13: Elinima as atualizações do dicionário rascunho D_R que foram inseridas devido às árvore $A(n_{2\,no+1})$ e $A(n_{2\,no+2})$.

Passo 14: Seta em $A(n_0)$ que as árvores $A(n_{2\,no+1})$ e $A(n_{2\,no+2})$ foram eliminadas.
Retorne $\left[\overline{B^j},\, A(n_0)\right]$.

Passo 15: Realize a concatenação dos blocos $\overline{B^{2j+1}}$ e $\overline{B^{2j+2}}$ da seguinte forma:

Se$(M > N)$ **entao** concatene na vertical, ou seja:

$$\overline{B^j} = \begin{pmatrix} \overline{B^{2j+1}} \\ \overline{B^{2j+2}} \end{pmatrix}$$

Se Não concatene na horizontal, ou seja:

$$\overline{B^j} = \begin{pmatrix} \overline{B^{2j+1}}\ \overline{B^{2j+2}} \end{pmatrix}$$

Passo 16: Atualize o dicionário D_R da seguinte forma:

Para $n = (0, 1, \ldots, w-1)$ **faça**

$$N_x = 2^{\lfloor 0.5(n+1) \rfloor}$$
$$M_x = 2^{\lfloor 0.5n \rfloor}$$
$$S_{N_x,\,M_x} = \{S_{N_x,\,M_x}\} \cup \left\{T^{N,M}_{N_x,M_x}\left[\overline{B^j}\right]\right\}$$

Passo 17: Atualize o custo do nó.

$$\{J_{no}\} = \left\{J_{(2\,no+1)} + J_{(2\,no+2)} + \lambda R_0(no)\right\}$$

Passo 18: **Retorne** $\left[\overline{B^j},\, A(n_0)\right]$.

Procedimento $\overline{B^j}$ = `Codifica`(N, M, B^j, no, $A(n_0)$)

Passo 1: **Se** ($A(n_{2\,no+1}) = A(n_{2\,no+2}) = 0$) **ou** (N=M=1) **então** segue para o **Passo 2**.

Se não vai para o **Passo 5**.

Passo 2: Localizar o índice i do bloco s_i dentro do subdicionário S_x com dimensão (N, M) (mesma de B^j) que melhor aproxima B^j segundo $J_{n_j} = D_{n_j} + \lambda R I_{n_j}$. Guarde i e faça $\overline{B^j} = s_i$.

Passo 3: **Se**($N = M = 1$) **então** guarde o índice i e retorne o bloco aproximado $\overline{B^j}$.

Se Não Siga para o passo 4.

Passo 4: Grave um *flag* igual a 1, o índice i e retorne o bloco $\overline{B^j}$.

Passo 5: Grave um *flag* igual a 0.

Se($M > N$) **entao** divida B^j, em dois blocos, B^{2j+1} e B^{2j+2}, mudando suas dimensões para $\frac{M}{2}$ e chame o procedimento de codificação na ordem:

$\overline{B^{2j+1}}$ = `Codifica`(N, $\frac{M}{2}$, B^{2j+1}, $2\,no+1$, $A(n_0)$)

$\overline{B^{2j+2}}$ = `Codifica`(N, $\frac{M}{2}$, B^{2j+2}, $2\,no+2$, $A(n_0)$)

Se Não divida B^j, em dois blocos, B^{2j+1} e B^{2j+2}, mudando suas dimensões para $\frac{N}{2}$ e chame o procedimento de codificação na ordem:

$\overline{B^{2j+1}}$ = `Codifica`($\frac{N}{2}$, M, B^{2j+1}, $2\,no+1$, $A(n_0)$)

$\overline{B^{2j+2}}$ = `Codifica`($\frac{N}{2}$, M, B^{2j+2}, $2\,no+2$, $A(n_0)$)

Passo 6: Realize a concatenação dos blocos $\overline{B^{2j+1}}$ e $\overline{B^{2j+2}}$ da seguinte forma:

Se($M > N$) **entao** concatene na vertical, ou seja:

$$\overline{B^j} = \begin{pmatrix} \overline{B^{2j+1}} \\ \overline{B^{2j+2}} \end{pmatrix}$$

Se Não concatene na horizontal, ou seja:

$$\overline{B^j} = \begin{pmatrix} \overline{B^{2j+1}} & \overline{B^{2j+2}} \end{pmatrix}$$

Passo 7: Atualize o dicionário D da seguinte forma:

Para $n = (0, 1, \ldots, w-1)$ **faça**

$$N_x = 2^{\lfloor 0.5(n+1) \rfloor}$$

$$M_x = 2^{\lfloor 0.5n \rfloor}$$

$$S_{N_x, M_x} = \{S_{N_x, M_x}\} \cup \left\{ T^{N,M}_{N_x, M_x} \left[\overline{B^j} \right] \right\}$$

A Inicialização do Dicionário não é realizada da mesma forma que no MMP. Para o MMP-RDI temos a dimensão 1x1 com uma quantidade definida de elementos, chamada de $NUMSEQ$, igual a 64. Tais elementos são igualmente espaçados de um valor definido pela variável *passo* e possuem valores pertencentes ao conjunto definido pelos níveis de cinza para imagens codificadas a 8 bits, ou seja de 0 até 255. Nesta implementação a faixa foi um pouco menor, variando de $\{0, \ldots, 250\}$. A partir da dimensão 1x1 usamos a transformação de escala para obtermos a inicialização para as outras dimensões. Esses valores foram obtidos para taxas pequenas, ou seja, abaixo de 1.2 bpp. Para taxas mais altas talvez valha a pena ter o dicionário com um número de elementos superiores. O algoritmo para essa inicialização pode ser visualizado abaixo:

Passo 1: Inicialize as seguintes variáveis

* $NUMSEQ = 64$
* $passo = \lfloor \frac{255}{NUMSEQ-1} \rfloor$

Passo 2: Inicialize o dicionário D da seguinte forma:

Para $n = (0, 1, \ldots, w-1)$ **faça**

$$N_x = 2^{\lfloor 0.5(n+1) \rfloor}$$
$$M_x = 2^{\lfloor 0.5n \rfloor}$$
$$S_{N_x, M_x} = \{S_{N_x, M_x}\} \cup \left\{T^{1,1}_{N_x, M_x}\left[n\, passo\right]\right\}$$

3.4 Resultados

Nesta sessão veremos todos os resultados para os algoritmos MMP, MMP-RDI e ainda, para efeito de comparação para o JPEG e o SPIHT. Esses resultados são compostos por curvas taxa-distorção onde a taxa R é dada em bits por pixel (bbp) e a distorção é dada pela $PSNR$ definidas na sessão 2.4. Os testes foram feitos com as imagens *lena, barbara, aerial, f16, gold, pp1205, pp1209*, todas elas no formato *pgm*, e com tamanho de 512×512. essas imagens podem ser obtidas enviando-se um e-mail para waldirjr@fucapi.br ou waldirsj@lps.ufrj.br. Abaixo temos algumas informações relevantes, são elas:

- Para o MMP e o MMP-RDI temos todas as imagens divididas em blocos de tamanho 16×16 (veja a figura 3.2) onde cada bloco é processado individualmente da esquerda para a direita e de cima para baixo.

- Os subdicionários possuem um tamanho de 100000 blocos. Nos testes realizados nenhuma imagem atingiu um subdicionário com tamanho superior a 100000 blocos, logo a probabilidade deste ser preenchido é baixa. Entretanto, esta limitação não é estrutural, outros valores podem ser facilmente obtidos.

- A inicialização dos dicionários tanto para o MMP quanto para o MMP-RDI encontram-se descritas respectivamente nas sessões 3.1 e 3.3.2 .

- A árvore de segmentação, composta por *flags* iguais a 0 e a 1 e os índices dos blocos escolhidos para a codificação possuem o passo de codificação por entropia executado através do codificador aritmético [26] com modelos diferentes para cada subdicionário.

- As figuras abaixo apresentam as curvas *taxa* × *distorção* para as imagens *aerial*, *barbara*, *lena*, *f16*, *gold pp1205* e *pp1209*. Para essas as imagens, as curvas para o MMP-RDI são superiores as do MMP e JPEG porém o algoritmo SPIHT apresenta melhores resultados.

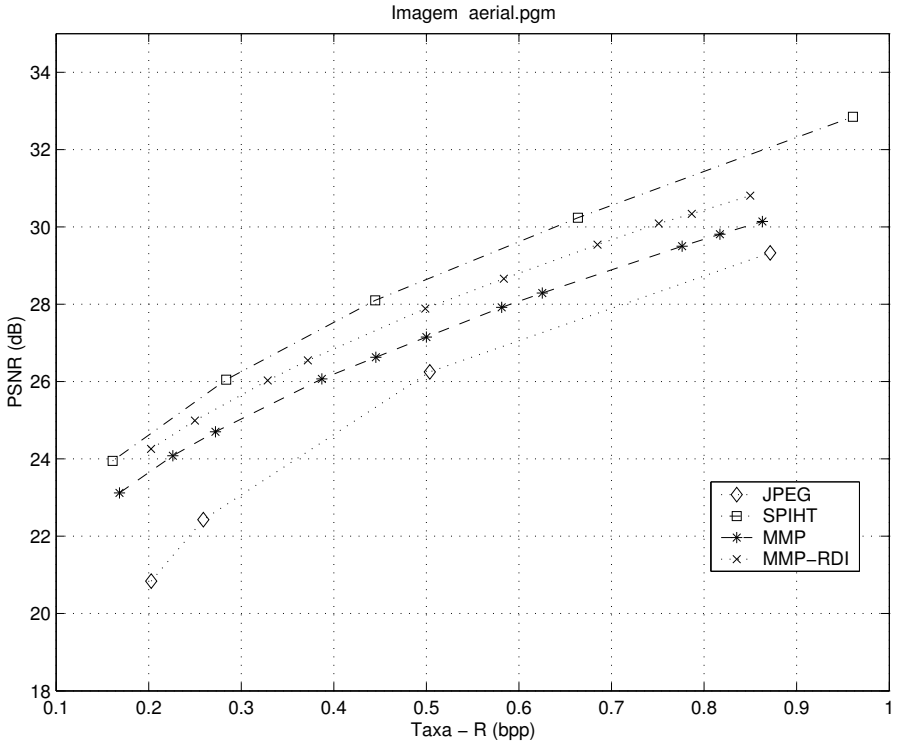


Figura 3.15: Curva taxa-distorção para a Imagem *aerial*

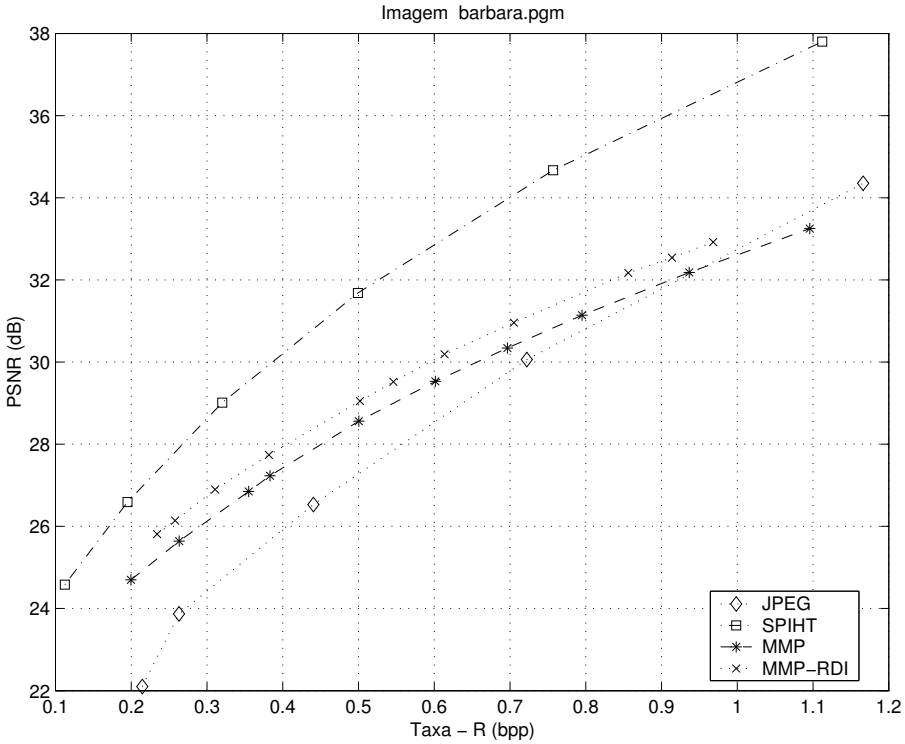


Figura 3.16: Curva taxa-distorção para a Imagem *barbara*

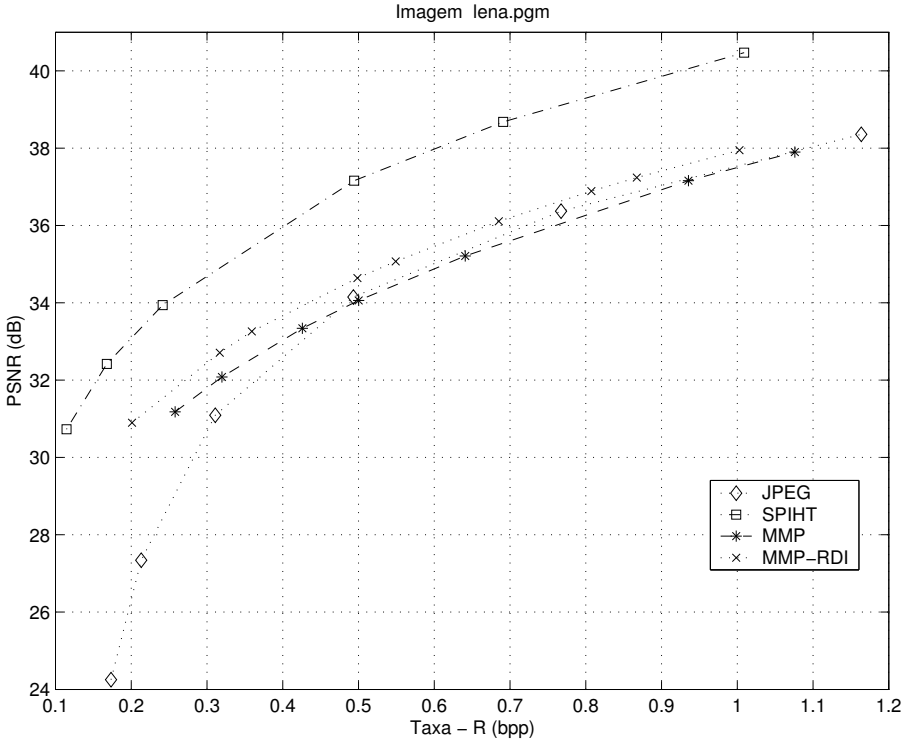


Figura 3.17: Curva taxa-distorção para a Imagem *lena*

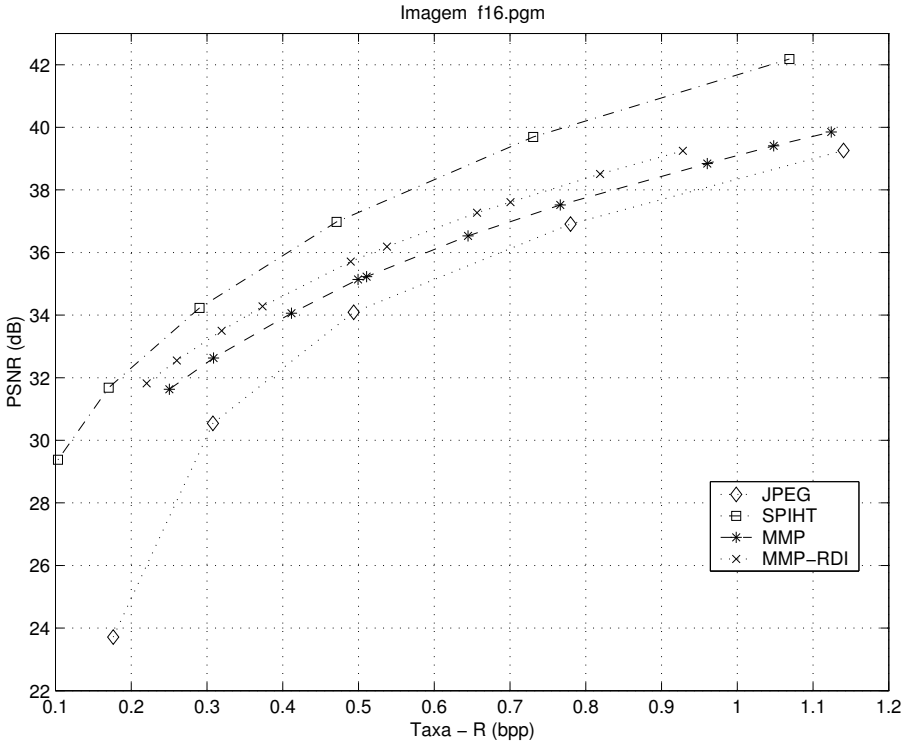


Figura 3.18: Curva taxa-distorção para a Imagem *f16*

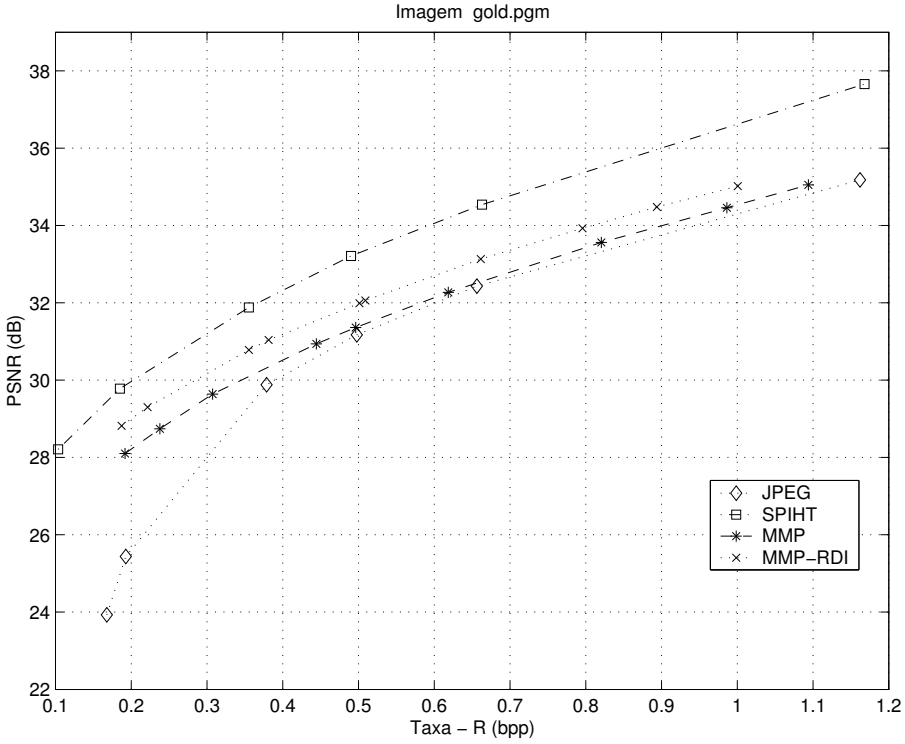


Figura 3.19: Curva taxa-distorção para a Imagem *gold*

- As figuras 3.20 e 3.21 apresentam as curvas *taxa* × *distorção* para as imagens *pp1205* e *pp1209*. Podemos perceber claramente o desempenho superior dos algoritmos baseados no MMP, pois estas imagens pertencem a classe de *imagens mistas*. Os melhores resultados, em ambas, são obtidos pelo MMP-RDI.

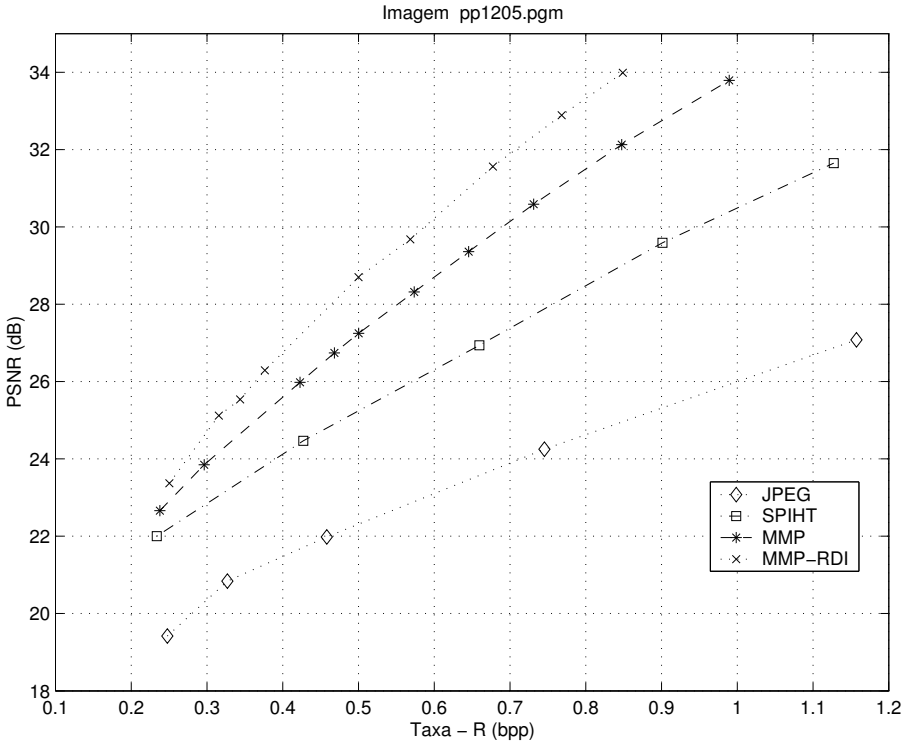


Figura 3.20: Curva taxa-distorção para a Imagem *pp1205*

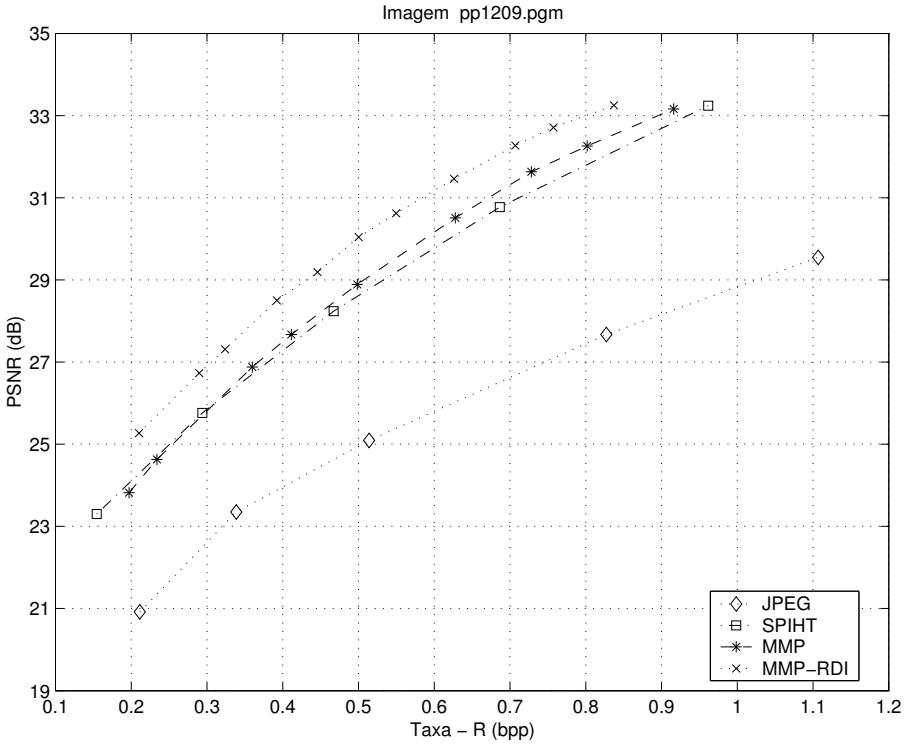


Figura 3.21: Curva taxa-distorção para a Imagem *pp1209*

3.5 Conclusão

Neste capítulo apresentamos o método de compressão de sinais chamado de *MMP*. O MMP é baseado na recorrência de padrões multiescala onde cada bloco do sinal de entrada é aproximado por um dicionário adaptativo sendo atualizado com versão dilatadas e contraídas de concatenações dos blocos previamente codificados. Seus melhores resultados são obtidos para classes de imagens mistas, ou seja, imagens que possuem texto e/ou sub-imagens. Nesta classe o MMP atinge por volta de 2,5dB de PSNR acima do método SPIHT, considerado estado da arte em compressão de imagens. Para imagens suaves o SPIHT apresenta resultados superiores.

De uma forma geral podemos concluir que o MMP possui as seguintes propriedades principais:

- **Universalidade**: O MMP não precisa de nenhum conhecimento prévio da imagem que será processada, isto significa que o seu comportamento pode

ser dito como universal. Logo nenhum paradigma é suposto incluindo-se o do domínio da frequência.

- **Dicionário Adaptativo**: O dicionário do MMP é atualizado com padrões diferentes à medida que a imagem é processada portanto seu dicionário é totalmente adaptativo.

- **Pode ser sem perdas**: O critério de controle do MMP pode ser ajustado de modo a podermos efetuar uma compressão sem perdas.

- **Desempenho**: Seu desempenho é excelente para imagens pertencentes a classe de imagens mistas, onde atingi-se resultados por volta de 2,5dB (PSNR) de ganho em relação a métodos considerados estado da arte em compressão de imagens. Para imagens suaves seus desempenho é inferior.

Capítulo 4

Método de União de Blocos (MMP-U1)

Neste capítulo começamos a apresentar o foco principal desta dissertação. Nosso objetivo é investigar sob quais circunstâncias o MMP pode efetuar *junções* ou *uniões* nos blocos. Desenvolvemos dois métodos que efetuam a união de blocos, o primeiro chama-se MMP-U1 e o segundo MMP-U3. A seguir iremos verificar a principal motivação para o algoritmo de união, a descrição do primeiro método (MMP-U1), os resultados obtidos e as conclusões.

4.1 Motivação

A motivação principal para o desenvolvimento da união de blocos para o MMP surgiu de [40], neste é proposto um novo método de codificação baseado em árvore binária, chamado *podagem-união* (*prune-join*) que consegue um comportamento exponencial da função taxa-destorção $R(D)$ para classes simples de sinais (conjunto de polinômios com um número finito de singularidades). Além disso, verifica-se que a função taxa-distorção do presente método é dada por $D(R) \sim c_0 2^{-c_1 R}$ (ver [40]), em contra partida, para codificadores baseados em wavelets a curva vale $D(R) \sim d_0\sqrt{R}\,2^{-d_1\sqrt{R}}$ (ver [40]). Dessa forma, a função $D(R)$ para o métode de podagem-união decresce mais rapidamente.

Esta seção divide-se em três partes. Na primeira parte examinamos o

método descrito em [40], na segunda mostramos os teoremas que regem a função taxa-distorção dos métodos e na última fazemos as conclusões.

4.1.1 O método podagem-união (prune-join)

O algoritmo proposto em [40], para o caso 1-D baseia-se na suposição de que temos uma função polinomial suave com um número finito de singularidadas. Neste caso, se segmentarmos tais funções então cada parte poderá ser representada por uma função polinomial. Considerando-se a função (ou sinal 1-D) abaixo, com dois polinômios e uma singularidade, podemos obter, através do algoritmo de *podagem* (*prune*), descrito no apêndice A.1 da dissertação, uma árvore binária da seguinte forma:

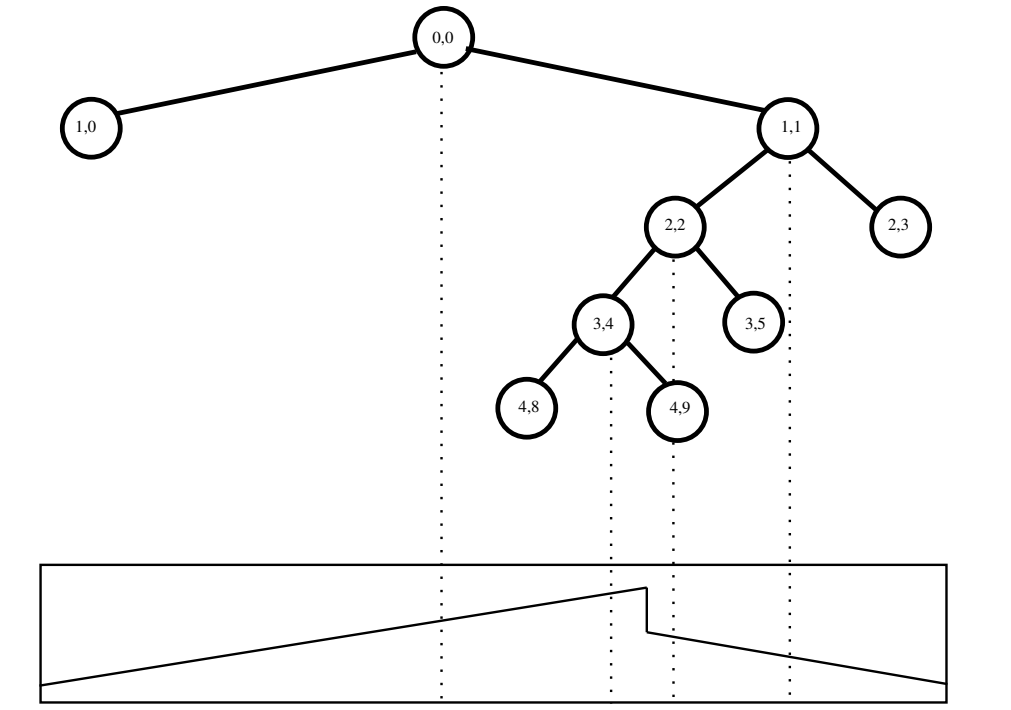


Figura 4.1: Algoritmo de Podagem

Nesta árvore podemos observar que não existe fusão (união) nos nós (2,3), (3,5), (4,9) e (1,0), (4,8) apesar de representarem a mesma informação pelo simples fato de possuirem nós pais diferentes. Isso pode ser visto na figura abaixo. Exatamente por este fato surge a necessidade de explorar *a dependência de nós folhas*

vizinhos que possuem nós pais diferentes. Tal necessidade é suprimida usando-se o método de *podagem-união* (*prune-join*).

O esquema de codificação *podagem-união*, fornece uma maneira de codificar nós folhas que carregam a mesma informação, dada pela união dos nós, desde que esta implique em um *custo menor*. Aqui, o *custo* é definido como o *Custo Lagrangiano* de cada nó folha considerado separadamente e de sua união. Consequentemente, dois nós folhas, com diferentes pais, podem ser unidos se o custo dessa união for menor ou igual que a soma dos custos dos nós computados separadamente, veja a figura 4.2.

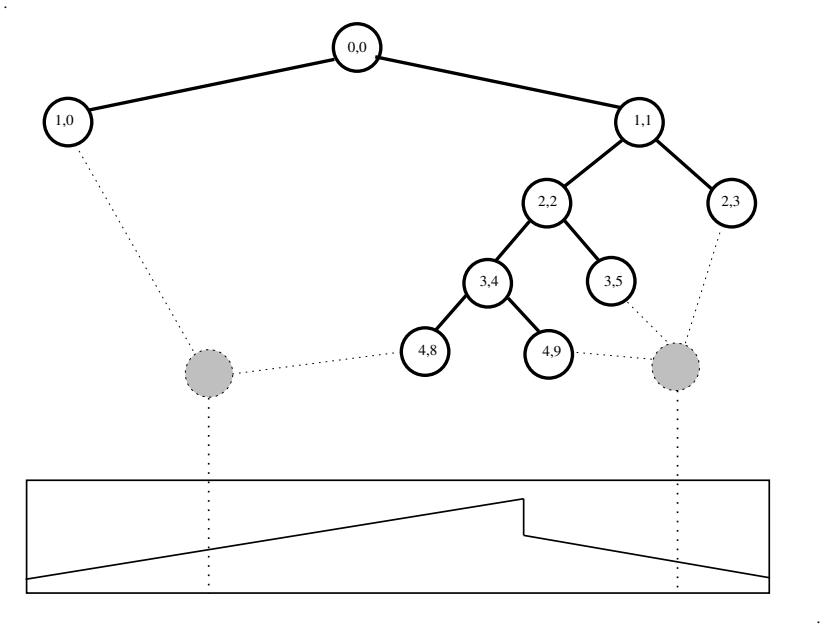


Figura 4.2: Algoritmo de Podagem-União

4.1.2 Comportamento taxa-distorção D(R)

Vamos considerar um sinal polinomial $f(t)$, definido no internalo $[0, T]$ que contém S singularidades. A análise taxa-distorção RD baseia-se, basicamente, em dois teoremas, apresentados por [40]. Esses teoremas podem ser enunciados da seguinte forma.

Teorema 4.1.1 (Função R-D: Algoritmo de Podagem) *O algoritmo de podagem com otimização RD, obtém o seguinte comportamento assintótico para a*

função taxa-distorção dado por:

$$D_P(R) \leq c_0\sqrt{R}\,2^{-c_1\sqrt{R}}. \tag{4.1}$$

Teorema 4.1.2 (Função R-D: Algoritmo de Podagem-União) *O algoritmo de podagem-união, que realiza a junção de nós vizinhos com pais diferentes, obtém um comportamento assintótico para a função taxa-distorção dado por:*

$$D_{PJ}(R) \leq c_2 2^{-c_3 R}. \tag{4.2}$$

■

É importante observar que o algoritmo de podagem é sub-ótimo no sentido taxa-distorção devido ao fato de codificar nós folhas que carregam a mesma informação. Já o algoritmo de podagem-união é ótimo nesse sentido pois permite a união dos nós que carregam a mesm informação. Outro fato bastante importante é que o teorema 4.1.1 fornece uma equação, para D_P que é função de $2^{-\sqrt{R}}$, ou seja, $D_P \propto 2^{-\sqrt{R}}$; em contra partida, a função taxa-distorção, dada pelo teorema 4.2, é função de 2^{-R}, ou seja, $D_P \propto 2^{-R}$, desta forma concluimos que, para uma mesma taxa R a distorção D é menor para o esquema de podagem-união.

4.1.3 Conclusão

Vimos que o método de podagem-união possui caracteristicas de desempenho superiores, para sinais polinomiais. Sabemos que o MMP possui uma estrutura semelhante à do método descrito acima, ou seja, ambos usam uma estrutura em árvore binária segmentada e seus custos são baseados no método de Lagrange. Diante disto somos levados a indagar a seguinte suposição:

A modificação do MMP incluindo o uso de uma estrutura podagem-união pode levar a ganhos de desempenho ?

Essa suposição, de uma maneira sintetizada, é o foco principal deste trabalho, nele investigamos sob quais circunstâncias o MMP (ver seção 3) pode contemplar uniões de blocos. A primeira proposta é o método chamado de **MMP-U1** o qual será descrito a seguir.

4.2 MMP-U1

4.2.1 Introdução

O MMP-U1, conforme mencionamos, é a primeira versão do algoritmo de união cujo objetivo principal é a junção de nós primos, ou seja, nós que não possuem o mesmo nó pai. Tal união deve gerar sempre um nó com dimensões quadradas ou retangulares e o seu custo lagrangeano deve ser menor que a soma dos custos dos nós computados separadamente. O custo lagrangeano possui algumas diferenças em relação ao descrito em [14, 15]. Para demonstrar o funcionamento do MMP-U1 vamos acompanhar o seguinte exemplo.

Considere uma árvore $A(n_0)$ e o seu bloco associado B dados a seguir:

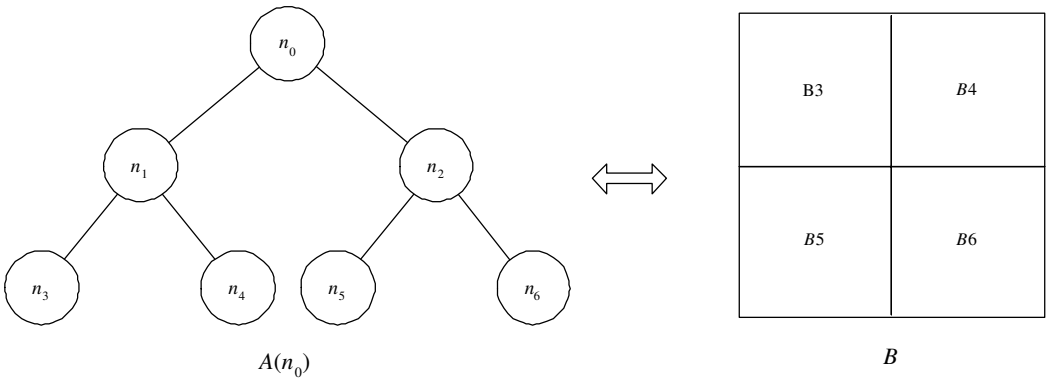


Figura 4.3: Árvore $A(n_0)$ e seu bloco associado B

É importante observar, antes de verificar o funcionamento do MMP-U1, que:

- É exatamente a partir do bloco associado B que as junções são avaliadas, por este motivo obrigatoriamente torna-se necessário o conhecimento prévio da estrutura da árvore $A(n_0)$, ou seja, a análise de união implica em um conhecimento prévio da árvore. Tal árvore é fornecida pela função `OtimizacaoRDI`, sem nenhuma modificação, descrita em 3.3.

- Existem duas informações sobre a união dos nós que precisamos considerar, são elas: Deve-se informar um *flag de junção* que indica se o bloco sofreu união ($flagjuncao = 1$) ou não ($flagjuncao = 0$) e um *flag de posição* que

indica a posição (horizontal ou vertical) de uma possível junção. Tais *flags* serão explicados com mais detalhes a seguir.

O MMP-U1 funciona da seguinte forma:

- A avaliação de união começa sempre do primeiro *nó folha*. Conforme o sentido de formação da árvore do MMP (ver capítulo 3), vemos que este nó, para a árvore $A(n_0)$ é igual à n_3, cujo bloco associado é $B3$, logo a avaliação de união começará por ele. Veja a figura 4.4 a seguir.

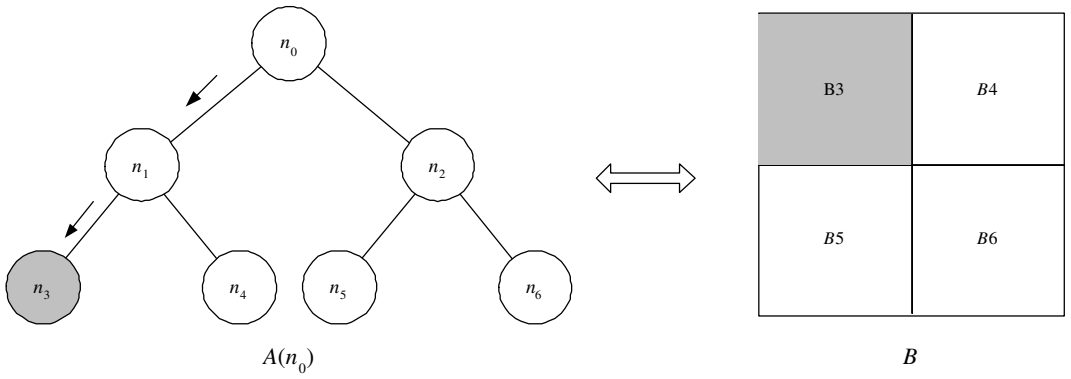


Figura 4.4: Árvore $A(n_0)$ e seu bloco associado B

- A principal característica do MMP-U1 é avaliar as uniões à direita e abaixo do bloco associado ao nó folha que está sendo considerado, ou seja, para o nosso exemplo os possíveis candidatos para a união são n_4 e n_5. Veja a figura 4.5.

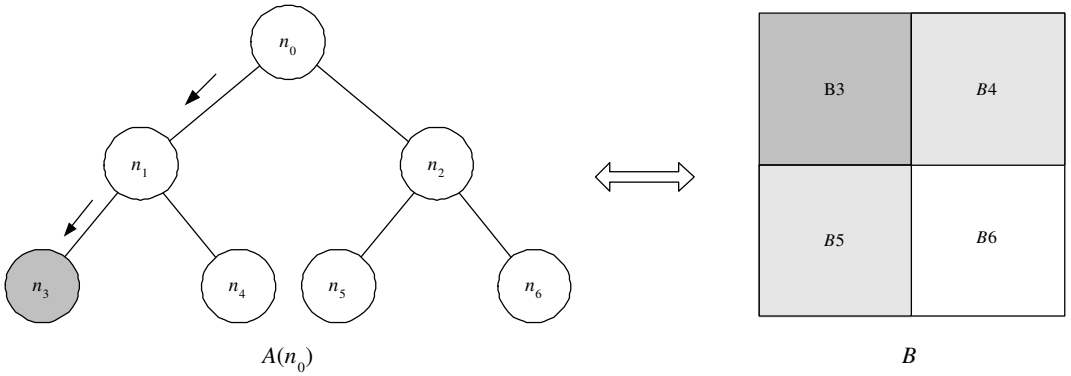


Figura 4.5: Análise para o nó folha n_3 e o bloco associado B

- O primeiro candidato, ou seja, n_4 não pode juntar com n_3 pois estes possuem o mesmo nó pai (n_1). Portanto resta apenas n_5, caso o custo da união $J_{J=n_{3,5}}$ seja menor que a soma dos custos computados separadamente $J_{n_3} + J_{n_5}$ então a união será confirmada. Supondo que a condição anterior seja satisfeita então temos a nossa primeira união dada por $n_{3,5}$, cujo bloco associado será $B_{J=3,5}$. Perceba que n_3 sofreu união com o de baixo e como só existe uma possível junção a informação da posição (na vertical) torna-se irrelevante. Veja a figura 4.6.

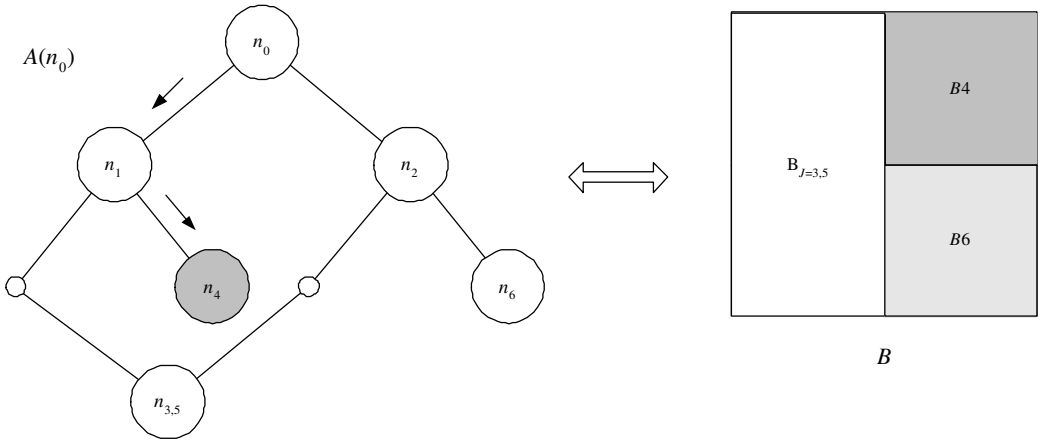


Figura 4.6: Análise para o nó folha n_3 e o bloco associado B

- A próxima folha é n_4. Esta possui apenas um candidato, n_6. Da mesma forma que no item anterior a união se confirma caso $J_{J=n_{4,6}} \leq J_{n_4} + J_{n_6}$. A posição não precisa ser informada pois estamos na mesma situação do item anterior. Veja a figura 4.7.

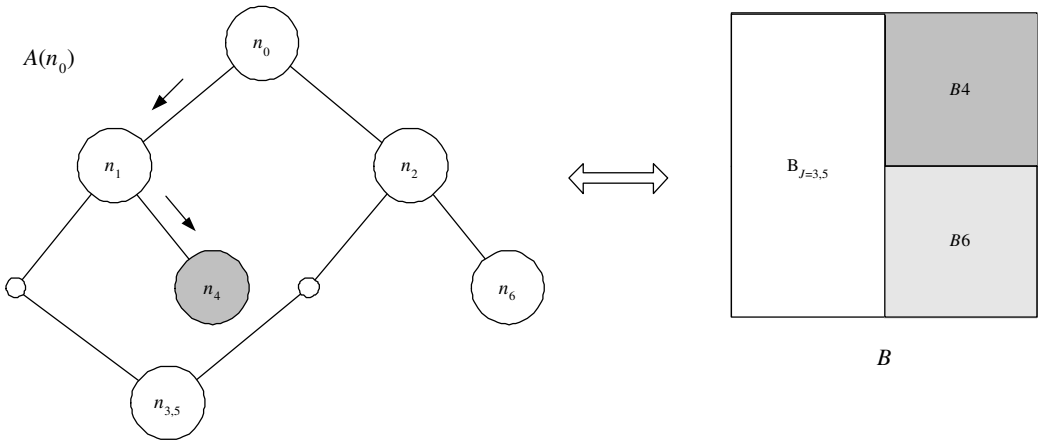


Figura 4.7: Análise para o nó folha n_4 e o bloco associado B

- A próxima folha seria n_5 porém esta já foi unida e portanto não será mais avaliada. O mesmo ocorre com n_6. Portanto, temos o seguite resultado:

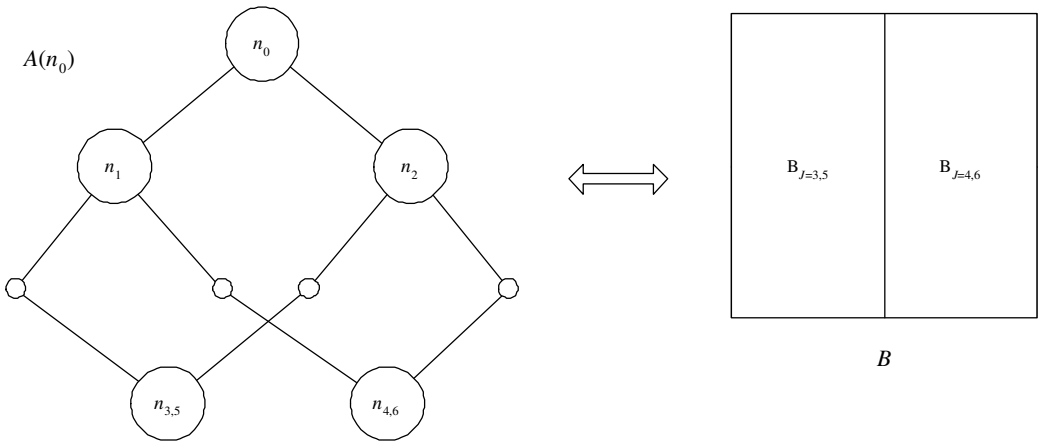


Figura 4.8: Árvore $A(n_0)$ Resultante e o bloco que será codificado B

- O bloco a ser codificado é apresentado na figura 4.8. Podemos fazer a comparação efetiva da economia de bits do MMP-U1 em relação ao MMP-RDI. Para o MMP-U1 foram usados 09 *flags* e 02 índices (veja a figura 4.9) e para o MMP-RDI foram usados 07 *flags* e 04 índices (veja a figura 4.10). Como sabemos, a taxa para um índice é muito maior que a taxa para um *flag*, portanto para nosso exemplo podemos afirmar que existe uma economia de aproximadamente dois índices.

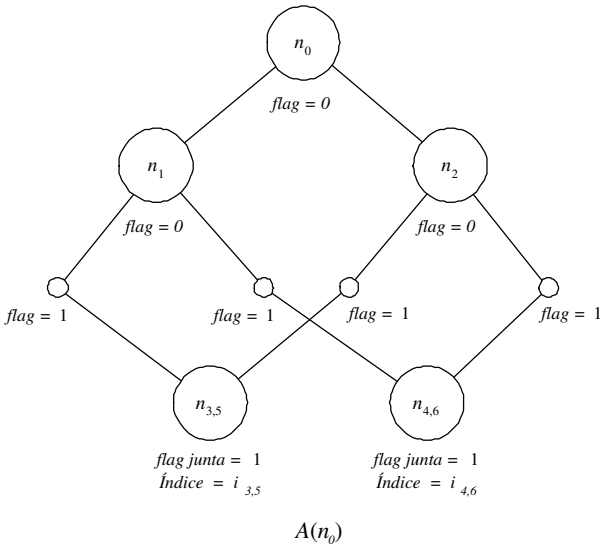


Figura 4.9: Árvore resultante para o MMP-U1

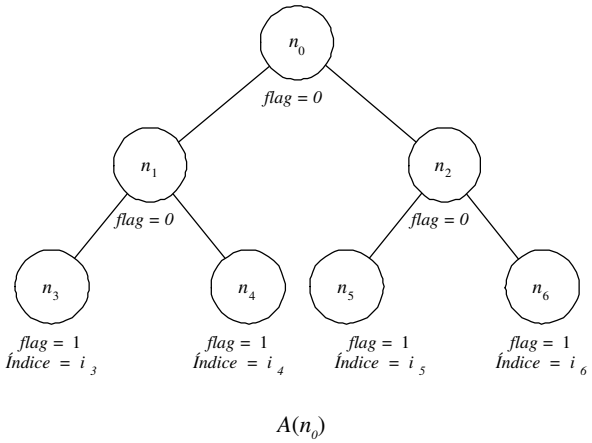


Figura 4.10: Árvore resultante para o MMP-RDI

■

Considere outro exemplo, conforme a figura 4.11, perceba que o *flag* de posição somente será utilizado quando existir a possibilidade de um nó se unir com outros dois nós (com o da direita e com o de baixo). Vamos desconsiderar a avaliação de união do nó n_{15} com n_{17} e ir direto para n_{16}, perceba que ele pode se unir tanto com o de baixo (n_{18}) quanto com o da direita (n_{19}). Somente uma união pode ser codificada ou seja ou temos $n_{16,18}$ ou $n_{16,19}$. Se decidirmos pela união de $n_{16,18}$ então o *flag* de posição vale 1 e se decidirmos por $n_{16,19}$ o *flag* de posição vale 0. Veja a figura 4.11.

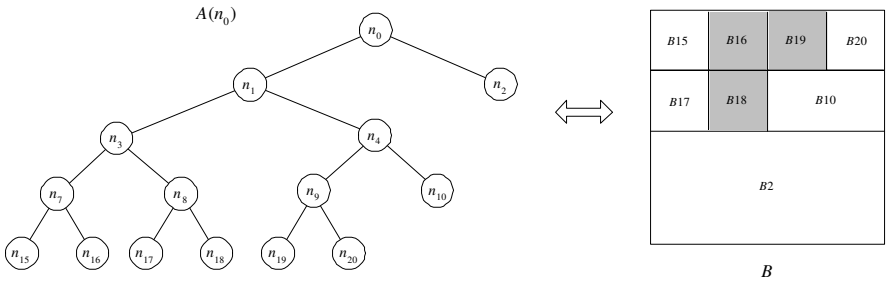


Figura 4.11: Uso do *flag* de posição

■

4.2.2 Particularidades do MMP-U1

A seguir iremos detalhar algumas particularidades do MMP-U1. A primeira delas é a quantidade de avaliações que devemos efetuar para um determinado nó, esta avaliação é limitada e definida através do teorema 4.2.1, a segunda é a inicialização do dicionário, pois existem algumas diferenças em relação ao MMP-RDI das quais serão comentadas e a última é o critério de decisão de junção.

A. Quantidade de Avaliações de União

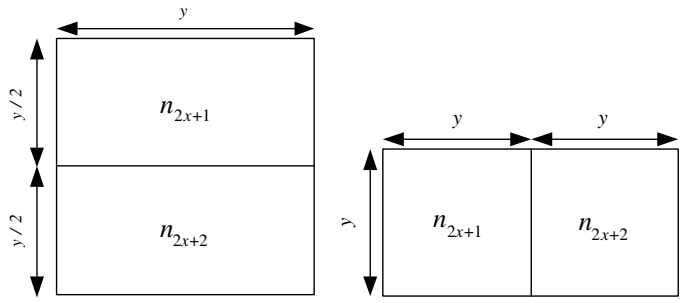
Existe uma relação intrínseca entre o número do nó folha ser par ou ímpar e o fato dele juntar somente com o da direita, somente com o de baixo ou com ambos. Tal ralação pode ser vista na forma de teorema como:

Teorema 4.2.1 (Uniões do MMP-U1) *Dada uma árvore $A(n_0)$ com um total de w nós folhas, um nó folha $n_k \in A(n_0)$ e o seu bloco associado Bk então temos as seguintes avaliações de união:*

(i) Se k é impar então existe somente uma avaliação a ser feita.
Se o bloco Bk for quadrado então a avaliação deve ser feita com o de baixo.
Se o bloco Bk for retangular então a avaliação deve ser feita com o da direita.

(ii) Se k é par então existem duas avaliações a serem feitas, tanto com o de baixo quanto com o da direita.

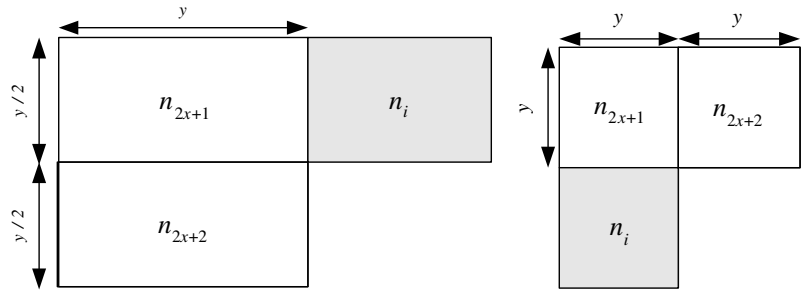
Prova Parte *(i)*: Considere, como na figura 4.12, um bloco associado particionado na horizontal e na vertical cujos nós folhas são n_{2x+1} e n_{2x+2} e seu nó pai é n_x com $x = \{0, 1, \ldots, 2^{\log(T)} - 1\}$, onde T é o valor máximo entre as dimensões horizontais ou verticais, ou seja, $T = 16$. De uma forma genérica temos que tanto na partição vertical quanto na horizontal o nó folha n_{2x+1}, que é ímpar, só pode juntar com um já que o nó n_{2x+2} possui o mesmo pai (n_x) de n_{2x+1}.



(a) Horizontal (b) Vertical

Figura 4.12: Partição de um bloco genérico

Se a partição for na horizontal (*colunas* > *linhas*) então temos apenas junção com o da direita, ou seja, temos uma junçao entre n_{2x+1} e n_i, veja na figura 4.13(a). Se a partição for na vertical (*colunas* = *linha*) então temos apenas junção com o de baixo, ou seja, pode ocorrer junção entre n_{2x+1} e n_i, veja figura 4.13(b).

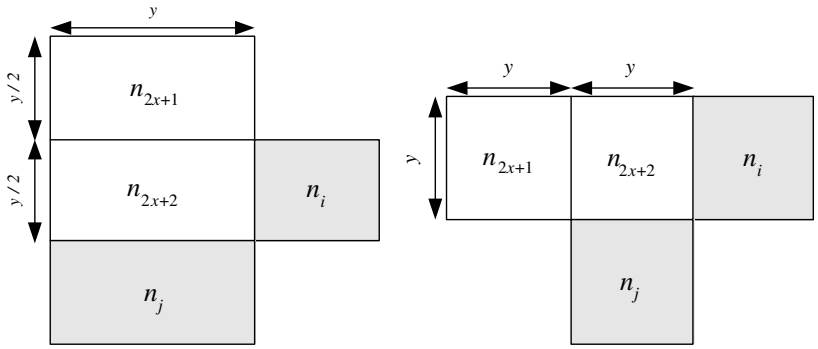


(a) Junção n_{2x+1} e n_i, à direita (b) Junção n_{2x+1} e n_i, abaixo

Figura 4.13: Junção para uma folha ímpar

Prova Parte *(ii)*: Tanto na partição vertical quanto na horizontal temos

sempre que o nó n_{2x+2} tem a possibilidade de unir com o da direita e com o de baixo já que esses nós não possuem o mesmo nó pai, veja a figura 4.14.



(a) Horizontal: junções com n_i e n_j

(b) Vertical: junções com n_i e n_j

Figura 4.14: Junção para uma folha par

A conclusão do teorema 4.2.1 e que não precisamos informar o *flag* de posição se o nó é ímpar pois só existe uma possibilidade de junção (abaixo e a direita). Tal *flag* deve ser informado apenas quando o bloco é par, isso resulta numa ecomomia de 1 *flag* de uma forma automática. Esse fato só acontece porque só efetuamos união com o de baixo e o da direita.

B. Inicialização do Dicionário

A principal modificação para a inicializacão dicionário deve-se ao fato de que existem dimensões de blocos diferentes das encontradas no MMP-RDI. Tais dimensões devem ser pré-computadas pois, quando ocorrer uma atualização esta dever ser feita em todas as dimensões. Antes vamos introduzir a seguinte notação:

- Considere que o conjunto de dicionários nas diversas escalas (MMP-RDI) era da forma $D = \{S_0, S_1, \ldots, S_w\}$, com $w = 2\log_2(T) + 1$, onde T é o valor máximo entre as dimensões horizontais ou verticais, ou seja, $T = 16$.

- Para o método de união (MMP-U1) o dicionário é igual a $D = \left\{S^{L,C}\right\}$, onde L é a dimensão horizontal e C é a dimensão vertical do subdicionário $S \in D$.

- Assim podemos obter o dicionário do método de união, ou seja, D através do algoritmo abaixo.

Procedimento $[D]$ = AdicionaNovasDimensoes(T, D, w)

Para $n = \{0, 1, \ldots, w-1\}$ **faça**

Para $m = \{0, 1, \ldots, w-1\}$ **faça**

***** Guarde em L_n, L_m a dimensão horizontal dos subdicionários posicionados em n e m. De acordo com a ordem da tabela 4.1.

***** Guarde em C_n, C_m a dimensão vertical dos subdicionários posicionados em n e m. De acordo com a ordem da tabela 4.1.

Se($L_n = L_m$ **e** $[C_n + C_m] \leq C_{max}$) **então**

Se o subdicionário $(S^{L_m, C_n + C_m}) \notin D$ **então** adicione este a D.

Se($C_n = C_m$ **e** $[L_n + L_m] \leq L_{max}$) **então**

Se o subdicionário $(S^{L_n + L_m, C_m}) \notin D$ **então** adicione este a D.

Fim Para

Fim Para Retorne D

Ao executarmos o algoritmo obtemos, incluindo os valores do MMP-RDI, um total de 21 subdicionários. Tais dimensões podem ser visualizadas na tabela 4.1. Todos os 21 subdicionários devem ser inicializados da mesma forma que o MMP-RDI.

Tabela 4.1: Subdicionários e suas Dimensões

subdicionário	Dimensão
00	(01,01)
01	(01,02)
02	(02,02)
03	(02,04)
04	(04,04)
05	(04,08)
06	(08,08)
07	(08,16)
08	(16,16)
09	(02,01)
10	(01,03)

subdicionário	Dimensão
11	(01,04)
12	(03,02)
13	(04,02)
14	(02,06)
15	(02,08)
16	(06,04)
17	(08,04)
18	(04,12)
19	(04,16)
20	(12,08)
21	(16,08)

C. Cálculo do Custo

Por conveniência, vamos relembrar o conceito de custo lagrangeano definido na seção 3.3.2; para isso devemos considerar uma sub-árvore $A(n_a)$ da figura 4.15 então o custo lagrangeano será dado, para cada nó, pelas equações abaixo.

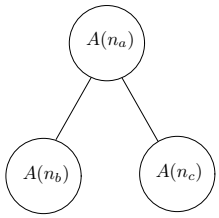


Figura 4.15: Sub-árvore $A(n_a)$

$$\begin{aligned} J_{n_a} &= D_{n_a} + \lambda RI_{n_a} \\ J_{n_b} &= D_{n_b} + \lambda RI_{n_b} \\ J_{n_c} &= D_{n_c} + \lambda RI_{n_c} \end{aligned} \tag{4.3}$$

Para o MMP-U1 temos uma configuração diferente que pode ser dividida em dois casos:

caso 1: Neste caso só temos uma avaliação a ser feita, ou seja, ou podemos unir com o da direita ou com o de baixo. Para exemplificar essa situação considere a árvore abaixo onde o nó atual é n_3 e o único nó à ser avaliado e n_5 (a baixo).

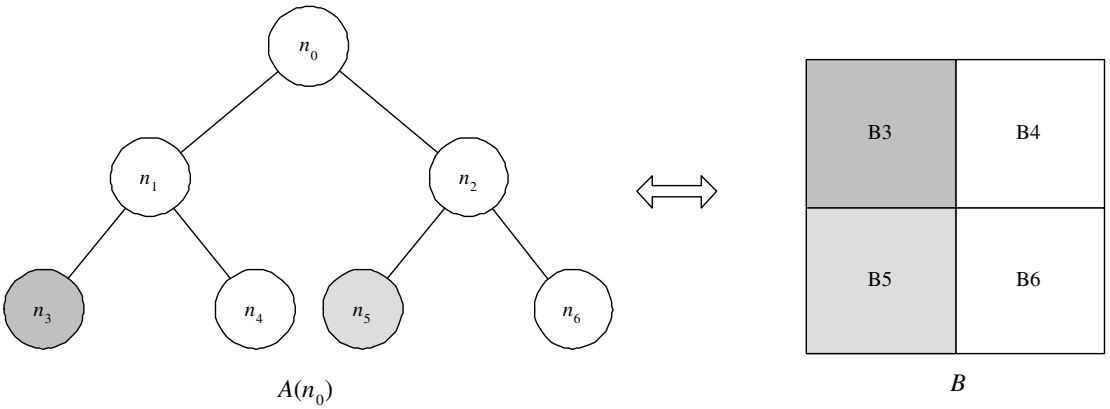


Figura 4.16: Caso 1

Para determinarmos se a união será confirmada ou não devemos avaliar os custos envolvidos, são eles:

$$\begin{aligned} J_{n_3} &= D_{n_3} + \lambda RI_{n_3} + \lambda R_{J0} \\ J_{n_5} &= D_{n_5} + \lambda RI_{n_5} + \lambda R_{J0} \\ J_{n_{3,5}} &= D_{n_{3,5}} + \lambda RI_{n_{3,5}} + \lambda R_{J1} \end{aligned}$$

Para este caso, a união somente será confirmada se:

$$\left\{J_{n_{3,5}}\right\} \leq \left\{J_{n_3} + J_{n_5}\right\}$$

Onde:

R_{J0}: indica a taxa para o *flag* de junção igual a 0. Este é considerado somente nos custos calculados separadamente pois o *flag* de junção 0 indica que os blocos não serão unidos.

R_{J1}: indica a taxa para o *flag* de junção igual a 1. Este é considerado apenas na união dos blocos pois é ele que indica sua união.

caso 2: Este é mais sofisticado, nele temos a possibilidade de juntar tanto com o da direita quanto com o de baixo. Perceba que existem duas situações distintas e dependendo de seus resultados teremos que tomar decisões diferentes.

Na *situação (a)*, descrita pela figura 4.17 avalia-se a união com o da direita com a diferença de que agora torna-se necessário informar a posição da união dada por $R_{P=0,1}$. A figura relacionada e os custos envolvidos são dados abaixo.

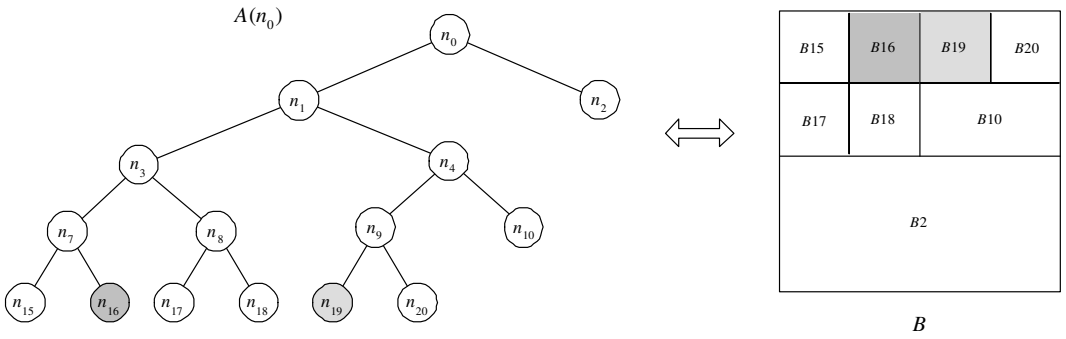


Figura 4.17: Caso 2: Avaliação com o da direita

$$
\begin{aligned}
J_{n16} &= D_{n16} + \lambda RI_{n16} + \lambda R_{J0} \\
J_{n19} &= D_{n19} + \lambda RI_{n19} + \lambda R_{J0} \\
J_{n16,19} &= D_{n16,19} + \lambda RI_{n16,19} + \lambda R_{J1} + \lambda R_{P0}
\end{aligned}
$$

Na *situação (b)* avalia-se a união com o de baixo sendo muito semelhante a primeira situação. Os custos envolvidos e a figura são mostrados a seguir.

$$
\begin{aligned}
J_{n16} &= D_{n16} + \lambda RI_{n16} + \lambda R_{J0} \\
J_{n18} &= D_{n18} + \lambda RI_{n18} + \lambda R_{J0} \\
J_{n16,18} &= D_{n16,18} + \lambda RI_{n16,18} + \lambda R_{J1} + \lambda R_{P1}
\end{aligned}
$$

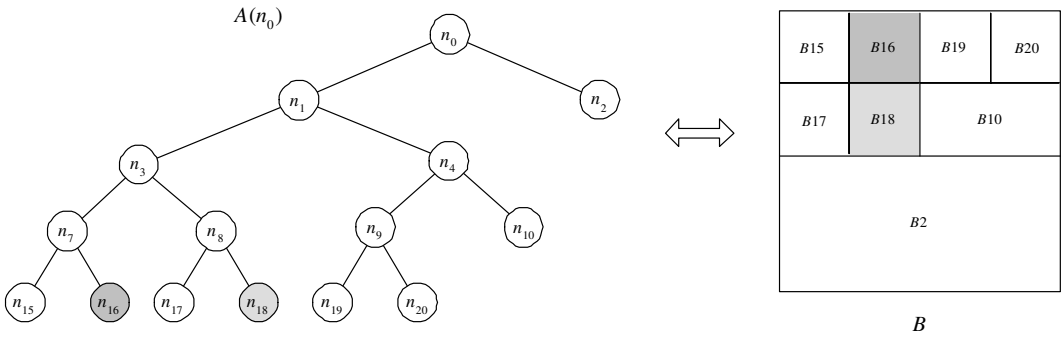


Figura 4.18: Caso 2: Avaliação com o de baixo

Uma outra forma de visualizarmos essas situações é mostrada através do algoritmo abaixo, inicialmente $caso2a = caso2b = 0$:

Se $\left(J_{n_{16,19}} \leq J_{n_{16}} + J_{n_{19}}\right)$ **então**
$caso2a = 1$
Se $\left(J_{n_{16,18}} \leq J_{n_{16}} + J_{n_{18}}\right)$ **então**
$caso2b = 1$

Dependendo do resultado do algoritmo acima devemos tomas as seguintes decisões:

Se ($caso2a = 1$ **e** $caso2b = 0$) **então**
Temos somente uma união portanto devemos confirmar a união $n_{16,19}$
Se ($caso2a = 0$ **e** $caso2b = 1$) **então**
Temos somente uma união portanto devemos confirmar a união $n_{16,18}$
Se ($caso2a = 1$ **e** $caso2b = 1$) **então**
Neste caso temos que decidir qual a junção que oferece a maior redução de custo, ou seja, confirmamos a união que possuir $\max\left\{J_{n_{16}} + J_{n_{19}} - J_{n_{16,19}}; J_{n_{16}} + J_{n_{18}} - J_{n_{16,18}}\right\}$

4.2.3 Algoritmo MMP-U1

O Objetivo do algoritmo de união é juntar blocos, que não possuem o mesmo nó pai, de modo que se obtenha um custo lagrangeano menor do que os dois blocos considerados separadamente.

É extremamente importante a ordem na qual as funções são chamadas. O primeiro passo a ser executado é a codificação dos *flags* de segmentação da árvore otimizada fornecida pela função `OtimizacaoRDI`, pois o decodificador precisa ser capaz de saber que um determinado bloco pode juntar somente com um para poder procurar pelo *flag* de junção.

O segundo passo seria a geração da tabela de união onde podemos encontrar todas as informações sobre possíveis uniões dos blocos. Quando a tabela de união estiver pronta podemos efetuar as avaliações de uniões, ou seja, decidimos quais são os blocos e uniões que serão efetivamente codificados.

O próximo passo é a codificação efetiva dos blocos, unidos ou não. Neste passo, são enviados os *flags* de junção, de posição e os índices do dicionário.

Portanto a ordem correta para o algoritmo de união de uma forma completa pode ser visualizada como na tabela 4.2.

Tabela 4.2: Ordem de Execução dos Procedimentos do MMP-U1 codificador

Ordem	Procedimento ou Função
1	`CodificaArvore()`
2	`GeraTabela()`
3	`OtimizacaoJuncao()`
4	`CodificaIndice()`

Na decodificação, também é extremamente importante a ordem na qual as funções são chamadas. Primeiramene realizamos a decodificação da árvore de segmentação. Após isto, realizamos a geração da tabela de forma idêntica ao codificador e finalmente decodificamos os índices dos blocos e se necessário suas posições. De uma forma completa o decodificado é dado pela tabela 4.3

Tabela 4.3: Ordem de Execução dos Procedimentos do MMP-U1 decodificador

Ordem	Procedimento ou Função
1	`DecodificaArvore()`
2	`GeraTabela()`
3	`DecodificaIndice()`

Logo é importante observar que as estatísticas dos *flags* de segmentação não são simultâneas as dos *flags* de junção, posição e os índices. Neste algoritmo criamos modelos para contabilizar as estatísticas dos *flags* de junção e posição. Para as estatísticas dos índices do dicionário basta realizarmos uma extensão para as novas dimensões. Veja a tabela 4.4.

Tabela 4.4: Adaptação do Modelo de frequências

Sigla	Comentários
f_fj	Frequência para o *flag* de junção. Este *flag* pode assumir o valor 1 indicando que a união deve ser feita e o valor 0 indicando o contrário.
f_fj_o	Frequência espelho (complemento) para o *flag* de junção.
f_fp	Frequência para o *flag* de posição. Este *flag* pode assumir o valor 1 indicando que a o bloco pode unir com o de baixo e o valor 0 indicando que é possível unir com o da direita.
f_fp_o	Frequência espelho (complemento) para o *flag* de posição.
f_i	Mesma descrição da tabela 3.1 com a única diferença de que este modelo é extendido para contemplar a contagem dos índices dos novos subdicionários. Os modelos espelho possuem a mesma notação adotado para o MMP-RDI com a adaptação da extensão.

A seguir apresentaremos os algoritmos de todas as funções que compõem o MMP-U1, seguindo a ordem descrita na tabela 4.2.

4.2.3.1 Função: `CodificaArvore`

Esta função segue o sentido de formação da árvore de segmentação do MMP. Ela é muito semelhante a função de codificação do MMP-RDI a única diferença é que iremos codificar apenas os *flags* de segmentação, e os índices não serão codificados. Sua saída fornece apenas um *flag* de segmentação e sua entrada é a árvore otimizada no sentido RD fornecida pela função `OtimizacaoRDI` $A(n_0)$ e a variável *no* indicando o nó que está sendo processado. Seu algoritmo completo é apresentado a seguir.

Procedimento `CodificaArvore`(N, M, no, $A(n_0)$)

Passo 1: **Se** $(A(n_{2\,no+1}) = A(n_{2\,no+2}) = 0)$ **ou** (N=M=1) **então** segue para o **Passo 2**.

Se não vai para o **Passo 3**.

Passo 2: Guarde um *flag* igual a 1.

Passo 3: Guarde um *flag* igual a 0.

Se$(M > N)$ **entao** chame a função `CodificaArvore` na ordem:

`CodificaArvore`(N, $\frac{M}{2}$, $2\,no+1$, $A(n_0)$)

`CodificaArvore`(N, $\frac{M}{2}$, $2\,no+2$, $A(n_0)$)

Se Não chame a função `CodificaArvore` na ordem:

`CodificaArvore`($\frac{N}{2}$, M, $2\,no+1$, $A(n_0)$)

`CodificaArvore`($\frac{N}{2}$, M, $2\,no+2$, $A(n_0)$)

4.2.3.2 Função: `GeraTabela`

Função responsável pela geração da tabela de união que contém informações imprescindiveis para realizarmos as avaliações de uniões dos blocos. Somente parte da tabela de união é preenchida pela função `GeraTabela` pois, até o momento deve-se informar basicamente se um nó pode unir com algum outro e se esta união ocorre com o bloco da direita ou com o de baixo ou com ambos. As informações contidas na tabela até então, serão usadas pela `OtimizacaoJuncao` para preencher sua parte restante.

A tabela de união é uma estrutura *tabela* composta pelos seguintes campos:

Tabela 4.5: Tabela de União e seus campos

num	campo	comentários
1	no	número do nó da árvore de segmentação, também pode ser associado com o bloco.
2	(N, M)	dimensão do bloco associado com o nó armazenado no campo no.
3	(nr, mr)	posicionamento do bloco associado com o nó em questão.
4	$flag\_junta$	indica se o nó foi unido, este campo é preenchido somente na função `OtimizacaoJuncao`. Se o nó foi unido seu valor é 1, caso contrário ele vale 0.
5	ja_foi_unido	Vale 1 se o nó já foi unido e 0 se não foi. Este campo indica à função `CodificaIndice` que o nó/bloco já foi unido não deve ser processado.
6	$juntar\_alguem$	Assume dois valores: 0 se o nó/bloco não pode juntar com ninguém e 1 se o nó/bloco pode juntar ou com o da direita ou com o de baixo.

num	campo	comentários
7	$juntar\_dois$	Caso nó possa junção tanto com o da dereita quanto com o de baixo seu valor é 1, caso contrário este campo vale 0.
8	$flag\_posicao$	Indica, caso o campo $juntar\_dois$ seja igual à 0, se o bloco pode juntar com o da direita (igual a 0) ou com o de baixo (igual a 1).
9	$(Nj1, Mj1)$	indica a dimensão das possíveis uniões.
10	$(Nj2, Mj2)$	indica a dimensão das possíveis uniões.
11	$juncao1$	indica o índice de uma possível união.
12	$juncao2$	indica o índice de uma possível união.

A função `GeraTabela` é composta por duas funções: A `GeraTabela1` que preenche os campos 1,2 e 3 e a `GeraTabela2` que preenche os campos $juntar\_dois$ e $juntar\_alguem$. O valor de $M_{max} = N_{max}$ é igual a 16. O acesso aos campos é dado segundo a seguinte notação: $tabela[indice] \cdot campo$.

Procedimento tabela = `GeraTabela`($A(n_0)$)

$ind = 0$

`GeraTabela1`(N_{max}, M_{max}, 0, 0, 0, $A(n_0)$);

$ind = 0$

`GeraTabela2`(N_{max}, M_{max}, 0, $A(n_0)$);

Procedimento `GeraTabela1`(N, M, nr, mr, no, $A(n_0)$);

Passo 1: **Se** ($A(n_{2\,no+1}) = A(n_{2\,no+2}) = 0$) **ou** (N=M=1) **então** segue para o **Passo 2**.

Se não vai para o **Passo 4**.

Passo 2: Preencha os seguintes campos:

$tabela[ind] \cdot no = no$

$tabela[ind] \cdot (N, M) = (N, M)$

$tabela[ind] \cdot (nr, mr) = (nr, mr)$

Passo 3: incremente o índice da tabela ind.

Passo 4: **Se** ($M > N$) **então** chame a função `GeraTabela1` na ordem:

`GeraTabela1`(N, $\frac{M}{2}$, nr, mr, $2\,no + 1$, $A(n_0)$)

`GeraTabela1`(N, $\frac{M}{2}$, nr, $mr + \frac{M}{2}$, $2\,no + 2$, $A(n_0)$)

Se Não chame a função `GeraTabela1` na ordem:

`GeraTabela1`($\frac{N}{2}$, M, nr, mr, $2\,no + 1$, $A(n_0)$)

`GeraTabela1`($\frac{N}{2}$, M, $nr + \frac{N}{2}$, mr, $2\,no + 2$, $A(n_0)$)

Procedimento `GeraTabela2`(N, M, no, $A(n_0)$);

Passo 1: **Se** ($A(n_{2\,no+1}) = A(n_{2\,no+2}) = 0$) **ou** (N=M=1) **então** segue para o **Passo 2**.

Se não vai para o **Passo 5**.

Passo 2: **Se**($no\,\%\,2 = 1$), ou seja, o nó é ímpar **então**

Se $M > N$ **então**

Observa se existe possibilidade de união com o bloco a direita verificando a igualdade dos vértices da direita do bloco analisado com os vértices da esquerda de algum dos blocos.

Se existir possibilidade **então** faça

Guarde o índice da tabela referente ao bloco que uniu à direita em m.

$tabela[ind] \cdot (Nj1, Mj1) = (tabela[ind].N,\ tabela[ind].M + tabela[m].M)$

$tabela[ind] \cdot juntar\_alguem = 1$

$tabela[ind] \cdot juntar\_dois = tabela[ind] \cdot flag\_posicao = 0$

$tabela[ind] \cdot juncao1 = m$

Se não

$tabela[ind].juntar\_alguem = tabela[ind].juntar\_dois = 0$

Se não

Observa se existe possibilidade de união com o bloco de baixo verificando a igualdade dos vértices de baixo do bloco analisado com os vértices de cima de algum dos blocos.

Se existir possibilidade **então** faça

Guarde o índice da tabela referente ao bloco que uniu com o de baixo em m.

$tabela[ind].(Nj1, Mj1) = (tabela[ind].N + tabela[m].N,\ tabela[ind].M)$

$tabela[ind].juntar\_alguem = 1$

$tabela[ind].juntar\_dois = 0$

$tabela[ind].flag\_posicao = 1$

$tabela[ind].juncao1 = m$

Se não

$tabela[ind].juntar\_alguem = tabela[ind].juntar\_dois = 0$

Passo 3: **Se**$(no \,\%\, 2 = 0)$, ou seja, o nó é par **então**

Verifica se existe possibilidade de união tanto com o da direita quanto com o de baixo.

Se ambas existirem **então**

$tabela[ind].juntar\_alguem = tabela[ind].juntar\_dois = 1$

Se existir apenas uma **então**

$tabela[ind].juntar\_alguem = 1$

$tabela[ind].juntar\_dois = 0$

Seta o campo $tabela[ind].flag\_posicao$ de acordo com o tipo (com o da direita ou de baixo)

Se não existir nenhuma **então**

$tabela[ind].juntar\_alguem = tabela[ind].juntar\_dois = 0$

Passo 4: incremente o índice da tabela *ind* e **Retorna**.

Passo 5: **Se**$(M > N)$ **entao** chame a função `GeraTabela2` na ordem:

`GeraTabela2`$(\ N,\ \frac{M}{2},\ 2\,no + 1,\ A(n_0)\ )$

`GeraTabela2`$(\ N,\ \frac{M}{2},\ 2\,no + 2,\ A(n_0)\ )$

Se Não chame a função `GeraTabela2` na ordem:

`GeraTabela2`$(\ \frac{N}{2},\ M,\ 2\,no + 1,\ A(n_0)\ )$

`GeraTabela2`$(\ \frac{N}{2},\ M,\ 2\,no + 2,\ A(n_0)\ )$

4.2.3.3 Função: `OtimizacaoJuncao`

É a função mais complexa do MMP-U1. Nela analisamos se as união possuem um custo lagrangeano menor que os blocos sozinhos e simulamos as estatísticas para os *flags* de junção, posição, e os índices do dicionário. Como a união segue uma orientação diferente da árvore de segmentação do MMP, os custos calculados para os blocos onde testa-se a união não são reais, no final deste capítulo (seção 4.3) comentaremos esse problema.

À medida que os nós vão sendo analisados, a tabela de união vai sendo preenchida, para, posteriormente codificarmos uma união ou os blocos separadamente.

Antes de chamar esta função devemos zerar os contadores de frequência espelho, ou seja, f_i_o, f_i_r, f_fj_o, f_fp_o e f_fs_o e o campo $tabela[ind] \cdot ja\_foi\_unido$ da tabela de união.

Procedimento $\overline{B^j}$ = `OtimizacaoJuncao`(N, M, B^j, no, $A(n_0)$, $tabela$);

Passo 1: **Se** $(A(n_{2\,no+1}) = A(n_{2\,no+2}) = 0)$ **ou** (N=M=1) **então** vai para o **Passo 2** caso contrário vai para o **Passo 7**

Passo 2: **Se** ($tabela[ind] \cdot ja\_foi\_unido = 1$) **então** faça incremente ind e **Retorne**.

Passo 3: **Se** ($tabela[ind] \cdot juntar\_alguem = 0$) **então**

3.1 Localizar o índice i do bloco s_i dentro dos subdicionários $S_x \in D$ com dimensão (N, M) (mesma de B^j) que melhor aproxima B^j segundo o custo $J_{n_j} = D_{n_j} + \lambda RI_{n_j}$. Guarde i e faça $\overline{B^j} = s_i$. O custo é dado por:

$$J_{n_j} = \left(s_i - B^j\right)^2 + \lambda \log_2 \frac{\sum fi + \sum fio + \sum fir}{fi + fio}$$

3.2 Verificar se algum bloco s_{i_R} dentro dos subdicionários $S_{Rx} \in D_R$ possuem uma aproximação melhor, em relação ao custo, que a escolhida no *3.1.* Se isso acontecer, guarde i e faça $\overline{B^j} = s_{i_R}$. O custo é dado por:

$$J_{n_j} = \left(s_i - B^j\right)^2 + \lambda \log_2 \frac{\sum fi + \sum fio + \sum fir}{fir}$$

3.3 **Se** (A aproximação foi feita por D) **então** Incrementa fio.
Se (A aproximação foi feita por D_R) **então** Incrementa fir.

3.4 Incremente o índice da tabela ind e **Retorne**.

Se não siga para o **Passo 4**

Passo 4: **Se** ($tabela[ind] \cdot juntar\_alguem = 1$) **então** siga para o **Passo 5**.
Se não Erro na Tabela.

Passo 5: **Se** ($tabela[ind] \cdot juntar\_dois = 0$) **então**

5.1 Localizar o índice i do bloco s_i dentro dos subdicionários $S_x \in D$ com dimensão (N, M) (mesma de B^j) que melhor aproxima B^j segundo o custo $J_{n_j} = D_{n_j} + \lambda RI_{n_j}$. Guarde i, e faça $\overline{Ba^j} = s_i$. O custo é dado pela equação abaixo, armazene este em J_{na}.

$$J_{n_j} = \left(s_i - B^j\right)^2 + \lambda \log_2 \frac{\sum fi + \sum fio + \sum fir}{fi + fio}$$

5.2 Verificar se algum bloco s_{i_R} dentro dos subdicionários $S_{Rx} \in D_R$ possuem uma aproximação melhor, em relação ao custo, que a escolhida no *5.1*. Se isso acontecer, guarde i, faça $\overline{Ba^j} = s_{i_R}$ e substitua valor de J_{na} pelo custo encontrado. O custo é dado por:

$$J_{n_j} = \left(s_i - B^j\right)^2 + \lambda \log_2 \frac{\sum fi + \sum fio + \sum fir}{fir}$$

5.3 Recupere, na tabela de união, o índice do candidato a unir com o nó atual e armazene em *tmp*.

5.4 Localize a melhor aproximação do bloco associado ao nó dado por $tabela[tmp] \cdot no$ nos dicionários D ou D_R segundo as equações citadas em *5.1* e *5.2*. Armazene o valor do custo encontrado em J_{nb}.

5.5 Localize a melhor aproximação do bloco associado ao nó união dado por $n_{na,nb}$ onde $na = tabela[tmp] \cdot no$ e $nb = tabela[ind] \cdot no$ nos dicionários D ou D_R segundo as equações citadas em *5.1* e *5.2*, armazene em $\overline{BJ^j}$ essa apromimação (dependendo da origem do dicionário). Armazene o índice em i_J e o valor do custo encontrado em J_{nJ}.

5.6 Calcule a taxa R_{J0} para o *flag* de junção igual a zero que será usado para para informar que o nó atual não sofre união e a taxa R_{J1} usada para informar que teremos união.

$$\lambda R_{J0} = \lambda \log_2 \frac{\sum f\_fj(0) + \sum f\_fj\_o(0)}{f\_fj(0) + f\_fj\_o(0)}$$

$$\lambda R_{J1} = \lambda \log_2 \frac{\sum f\_fj(1) + \sum f\_fj\_o(1)}{f\_fj(1) + f\_fj\_o(1)}$$

5.7 Calcule a taxa R_{J0b} para o bloco condidato a união. Este cálculo somente deve ser efetuado se esse bloco puder unir com alguém, ou seja, $tabela[tmp] \cdot juntar\_alguem = 1$, caso contrário este valor é igual a zero.

$$\lambda R_{J0b} = \lambda \log_2 \frac{\sum f\_fj(0) + \sum f\_fj\_o(0) + 1}{f\_fj(0) + f\_fj\_o(0) + 1}$$

5.8 Faça a avaliação de união, ou seja:

Se $\{J_{nJ} + \lambda R_{J1}\} \leq \{J_{na} + \lambda R_{J0} + J_{nb} + \lambda R_{J0b}\}$ **então** vai para *5.9*

Se não siga para o *5.10*

5.9 Faça $\overline{B^j} = \overline{BJ^j}$, incremente o *flag* de junção $f\_fs\_o(1)$ e o índice i_J como:

Se (A aproximação $\overline{BJ^j}$ foi feita por D) **então** incremente fio.

Se (A aproximação $\overline{BJ^j}$ foi feita por D_R) **então** incremente fir.

$tabela[ind] \cdot flag\_junta = 1$

$tabela[tmp] \cdot ja\_foi\_unido = 1$

Guade o *flag* referente à posição em $tabela[ind] \cdot flag\_posicao$

Vá para o *5.11*

5.10 Faça $\overline{B^j} = \overline{Ba^j}$, incremente o *flag* de junção $f\_fs\_o(0)$ e o índice i como:

Se (A aproximação $\overline{Ba^j}$ foi feita por D) **então** incremente fio.

Se (A aproximação $\overline{Ba^j}$ foi feita por D_R) **então** incremente fir.

$tabela[ind] \cdot flag\_junta = 0$

Vá para o *5.11*

5.11 Incremente o índice da tabela ind e **Retorne** $\overline{B^j}$.

Se não siga para o **Passo 6**

Passo 6: **Se** ($tabela[ind] \cdot juntar\_dois = 1$) **então**

6.1 Localizar o índice i de forma análoga aos itens *5.1 e 5.2* que melhor aproxima B^j. Guardar o valor de i, a aproximação em $\overline{Ba^j}$ e o custo em J_{na}.

6.2 Recuperar os índices da tabela dos dois candidatos a junção e armazenar em $tmp1$ e $tmp2$.

6.3 Localizar o índice i de forma análoga aos itens *5.1 e 5.2* que melhor aproxima o bloco associado ao nó $n_b = tabela[tmp1] \cdot no$. Calcular o custo e armazenar em J_{nb}.

6.4 Calcular as taxas para os *flags* de posição e de junção da seguinte forma:

Se $tabela[tmp1] \cdot flag\_posicao = 0$

$$\lambda R_P = \lambda \log_2 \frac{\sum f\_fp(0) + \sum f\_fp\_o(0)}{f\_fp(0) + f\_fp\_o(0)}$$

Se $tabela[tmp1] \cdot flag\_posicao = 1$

$$\lambda R_P = \lambda \log_2 \frac{\sum f\_fp(1) + \sum f\_fp\_o(1)}{f\_fp(1) + f\_fp\_o(1)}$$

Se $tabela[tmp1] \cdot juntar\_alguem = 0$ **então** $R_{J0b} = 0$ caso contrário

$$\lambda R_{J0b} = \lambda \log_2 \frac{\sum f\_fj(0) + \sum f\_fj\_o(0)}{f\_fj(0) + f\_fj\_o(0)}$$

A taxa para a união ou não é dada, para o bloco atual por:

$$\lambda R_{J0} = \lambda \log_2 \frac{\sum f\_fj(0) + \sum f\_fj\_o(0)}{f\_fj(0) + f\_fj\_o(0)}$$

$$\lambda R_{J1} = \lambda \log_2 \frac{\sum f\_fj(1) + \sum f\_fj\_o(1)}{f\_fj(1) + f\_fj\_o(1)}$$

6.5 Localizar o índice i_{J1} de forma análoga aos itens *5.1 e 5.2* que melhor aproxima o bloco associado ao nó união dado por $n_{na,nb}$ onde $na = tabela[ind] \cdot no$ e $nb = tabela[tmp1] \cdot no$, armazene em $\overline{(BJ1)^j}$ essa apromimação (dependendo da origem do dicionário). Armazene o índice em i_{J1} e o valor do custo encontrado em $CJ1$.

6.6 Avalie a primeira união, armazenando o resultado em u_1 ou seja:

Se $\{CJ1 + \lambda R_{J1} + \lambda R_P\} \leq \{J_{na} + \lambda R_{J0} + J_{nb} + \lambda R_{J0b}\}$ **então** $u_1 = 1$

Se não $u_1 = 0$

Calcule a redução $RED1 = (J_{na} + J_{nb}) - CJ1$

6.7 Repita os passos *6.3* e *6.4* com o índice da tabela $tmp2$. Repita o passo *6.5*, armazene em $\overline{(BJ2)^j}$ a apromimação, armazene o índice em i_{J2} e o valor do custo encontrado em $CJ2$.

6.8 Avalie a segunda união, armazenando o resultado em u_2 ou seja:

Se $\{CJ2 + \lambda R_{J1} + \lambda R_P\} \leq \{J_{na} + \lambda R_{J0} + J_{nb} + \lambda R_{J0b}\}$ **então** $u_2 = 1$

Se não $u_2 = 0$

Calcule a redução $RED2 = (J_{na} + J_{nb}) - CJ2$

Passo 7: **Se** $(u_1 = 0$ **e** $u_2 = 0)$ **então**

Vá para o **Passo 8**.

Se $(u_1 = 1$ **e** $u_2 = 0)$ **ou** $(u_1 = 0$ **e** $u_2 = 1)$ **então**

Vá para o **Passo 9**.

Se ($u_1 = 1$ **e** $u_2 = 1$) **então**

Se $RED1 > RED2$ **então** faça $u_2 = 0$ e vá para o **Passo 9**.

Se não faça $u_1 = 0$ e vá para o **Passo 9**.

Passo 8: Faça $\overline{B^j} = \overline{Ba^j}$, incremente o *flag* de junção $f\_fs\_o(0)$ e o índice i como:

Se (A aproximação $\overline{Ba^j}$ foi feita por D) **então** incremente fio.

Se (A aproximação $\overline{Ba^j}$ foi feita por D_R) **então** incremente fir.

$tabela[ind] \cdot flag\_junta = 0$

Incremente o índice da tabela ind e **Retorne** $\overline{B^j}$.

Passo 9: **Se** $u_1 = 1$ **então** vá para *9.1*.

Se não vá para *9.2*.

9.1 Faça $\overline{B^j} = \overline{BJ1^j}$ e $i_J = i_{J1}$.

$tabela[tmp1] \cdot ja\_foi\_unido = 1$

Incremente o índice i_{J1} como:

Se (A aproximação $\overline{BJ1^j}$ foi feita por D) **então** incremente fio.

Se (A aproximação $\overline{BJ1^j}$ foi feita por D_R) **então** incremente fir.

Vá para *9.3*.

9.2 Faça $\overline{B^j} = \overline{BJ2^j}$ e $i_J = i_{J2}$.

$tabela[tmp2] \cdot ja\_foi\_unido = 1$

Incremente o índice i_{J2} como:

Se (A aproximação $\overline{BJ2^j}$ foi feita por D) **então** incremente fio.

Se (A aproximação $\overline{BJ2^j}$ foi feita por D_R) **então** incremente fir.

9.3 Incremente o *flag* de junção $f\_fs\_o(1)$

$tabela[ind] \cdot flag\_junta = 1$

Guade o *flag* referente à posição em $tabela[ind] \cdot flag\_posicao$

Incremente o índice da tabela ind e **Retorne** $\overline{B^j}$.

Passo 10: **Se**($M > N$) **entao** chame a função `OtimizacaoJuncao` na ordem:

$\overline{B^{2j+1}}$ = `OtimizacaoJuncao`(N, $\frac{M}{2}$, $2\,no + 1$, $A(n_0)$, $tabela$)

$\overline{B^{2j+2}}$ = `OtimizacaoJuncao`(N, $\frac{M}{2}$, $2\,no + 2$, $A(n_0)$, $tabela$)

Se Não chame na ordem:

$\overline{B^{2j+1}} = \texttt{OtimizacaoJuncao}(\ \frac{N}{2},\ M,\ 2\,no+1,\ A(n_0),\ tabela\ )$

$\overline{B^{2j+2}} = \texttt{OtimizacaoJuncao}(\ \frac{N}{2},\ M,\ 2\,no+2,\ A(n_0),\ tabela\ )$

Passo 11: Atualize o dicionário D_R da seguinte forma:

Para $n = (0, 1, \ldots, w-1)$ **faça**

$N_x = 2^{\lfloor 0.5(n+1) \rfloor}$

$M_x = 2^{\lfloor 0.5n \rfloor}$

$S_{N_x,\,M_x} = \{S_{N_x,\,M_x}\} \cup \left\{T^{N,M}_{N_x,M_x}\left[\overline{B^j}\right]\right\}$

4.2.3.4 Função: `CodificaIndice`

Nesta realizamos a codificação dos blocos. Se um bloco foi marcado como unido então codificamos a união caso contrário codificamos o bloco atual.

Procedimento $\overline{B^j}$ = `CodificaIndice`(N, M, B^j, no, $A(n_0)$, $tabela$)

Passo 1: **Se** ($A(n_{2\,no+1}) = A(n_{2\,no+2}) = 0$) **ou** (N=M=1) **então** segue para o **Passo 2**.

Se não vai para o **Passo 7**.

Passo 2: **Se** $tabela[ind] \cdot ja\_foi\_unido = 1$ faça o incremento de ind e **Retorne**

Se não siga para o **Passo 3**

Passo 3: **Se** ($tabela[ind] \cdot flag\_junta = 0$ **e** $tabela[ind] \cdot juntar\_alguem = 0$) **então**

3.1 Localizar o índice i do bloco s_i dentro do subdicionário S_x com dimensão (N, M) (mesma de B^j) que melhor aproxima B^j segundo $J_{n_j} = D_{n_j} + \lambda RI_{n_j}$. Guarde i e faça $\overline{B^j} = s_i$.

3.2 Guarde o índice i, incremente o índice da tabela (ind) e retorne o bloco $\overline{B^j}$.

Passo 4: **Se** ($tabela[ind] \cdot flag\_junta = 0$ **e** $tabela[ind] \cdot juntar\_alguem = 1$) **então**

4.1 Guarde o *flag* de junção igual a 0.

4.2 Localizar o índice i do bloco s_i dentro do subdicionário S_x com dimensão (N, M) (mesma de B^j) que melhor aproxima B^j segundo $J_{n_j} = D_{n_j} + \lambda RI_{n_j}$. Guarde i e faça $\overline{B^j} = s_i$.

4.3 Guarde o índice i, faça o incremento de ind e retorne o bloco $\overline{B^j}$.

Passo 5: **Se** ($tabela[ind] \cdot flag\_junta = 1$ **e** $tabela[ind] \cdot juntar\_alguem = 1$ **e** $tabela[ind] \cdot juntar\_dois = 0$) **então**

5.1 Guarde o *flag* de junção igual a 1.

5.2 Recupere o bloco que vai unir na tabela de união.

5.3 Localizar o índice i do bloco s_i dentro do subdicionário S_x com dimensão do bloco união obtido em *5.2* segundo $CJ_n = D_n + \lambda RI_n$ onde n é o nó junção associado ao bloco união, guarde i e faça $\overline{B^j} = s_i$.

5.4 Guarde o índice i, faça o incremento de ind e retorne o bloco $\overline{B^j}$.

Passo 6: **Se** ($tabela[ind] \cdot flag\_junta = 1$ **e** $tabela[ind] \cdot juntar\_alguem = 1$ **e** $tabela[ind] \cdot juntar\_dois = 1$) **então**

6.1 Guarde o *flag* de junção igual a 1.

6.2 Recupere o bloco que vai unir na tabela de união.

6.3 Guarde o *flag* de posição de acordo com a união obtida (à direita igual a 0 ou com o de baixo igual a 1).

6.4 Localizar o índice i do bloco s_i dentro do subdicionário S_x com dimensão do bloco união obtido em *6.2* segundo $CJ_n = D_n + \lambda RI_n$ onde n é o nó junção associado ao bloco união, guarde i e faça $\overline{B^j} = s_i$.

6.5 Guarde o índice i, faça o incremento de ind e retorne o bloco $\overline{B^j}$.

Passo 7: **Se**$(M > N)$ **entao** divida B^j, em dois blocos, B^{2j+1} e B^{2j+2}, mudando suas dimensões para $\frac{M}{2}$ e chame o procedimento na ordem:
$\overline{B^{2j+1}} =$ `CodificaIndice`(N, $\frac{M}{2}$, B^{2j+1}, $2\,no+1$, $A(n_0)$, $tabela$)
$\overline{B^{2j+2}} =$ `CodificaIndice`(N, $\frac{M}{2}$, B^{2j+2}, $2\,no+2$, $A(n_0)$, $tabela$)
Se Não divida B^j, em dois blocos, B^{2j+1} e B^{2j+2}, mudando suas dimensões para $\frac{N}{2}$ e chame o procedimento de codificação na ordem:
$\overline{B^{2j+1}} =$ `CodificaIndice`($\frac{N}{2}$, M, B^{2j+1}, $2\,no+1$, $A(n_0)$)
$\overline{B^{2j+2}} =$ `CodificaIndice`($\frac{N}{2}$, M, B^{2j+2}, $2\,no+2$, $A(n_0)$, $tabela$)

Passo 8: Realize a concatenação dos blocos $\overline{B^{2j+1}}$ e $\overline{B^{2j+2}}$ da seguinte forma:
Se$(M > N)$ **entao** concatene na vertical, ou seja:

$$\overline{B^j} = \begin{pmatrix} \overline{B^{2j+1}} \\ \overline{B^{2j+2}} \end{pmatrix}$$

Se Não concatene na horizontal, ou seja:

$$\overline{B^j} = \begin{pmatrix} \overline{B^{2j+1}} & \overline{B^{2j+2}} \end{pmatrix}$$

Passo 9: Atualize o dicionário D da seguinte forma:
Para $n = (0, 1, \ldots, w-1)$ **faça**

$$N_x = 2^{\lfloor 0.5(n+1) \rfloor}$$

$$M_x = 2^{\lfloor 0.5n \rfloor}$$

$$S_{N_x,\,M_x} = \{S_{N_x,\,M_x}\} \cup \left\{ T^{N,M}_{N_x,M_x} \left[\overline{B^j} \right] \right\}$$

4.2.3.5 Função: `DecodificaArvore`

Esta é a primeira função do decodificador. Ela funciona de forma semelhante a função de decodificação do MMP original. Sua diferença é que não decodifica-se os índices dos blocos, ou seja, decodificamos apenas os *flags* de segmentação. A estrutura $A \cdot no$ indica a árvore de segmentação decodificada.

Procedimento $A =$ `DecodificaArvore`$(\ N,\ M,\ no\ )$

Passo 1: **Se** $(N = M = 1)$ **então** $A \cdot no = 1$.

Passo 3: Leia um *flag* e armazene em $varflag$. **Se** $(varflag = 1)$ **então** faça

$A \cdot (2\,no + 1) = 0.$

$A \cdot (2\,no + 2) = 0.$

$A \cdot no = 1.$

Se não faça $A \cdot no = 1$ e siga para o passo 4.

Passo 4: **Se**$(M > N)$ **entao** faça

$A =$ `Decodifica`$(\ N,\ \frac{M}{2},\ 2\,no + 1\ )$

$A =$ `Decodifica`$(\ N,\ \frac{M}{2},\ 2\,no + 2\ )$

Se Não faça

$A =$ `Decodifica`$(\ \frac{N}{2},\ M,\ 2\,no + 1\ )$

$A =$ `Decodifica`$(\ \frac{N}{2},\ M,\ 2\,no + 2\ )$

4.2.3.6 Função: `DecodificaIndice`

Esta realiza a decodificação dos índices dos blocos, e se necessário, os *flags* de posição.

Procedimento $\overline{B^j}$ = `DecodificaIndice`(N, M, no, $A(n_0)$, $tabela$)

Passo 1: **Se** $(A(n_{2\,no+1}) = A(n_{2\,no+2}) = 0)$ **ou** $(N = M = 1)$ **então**
vai para o **Passo 2**.
Se não vai para o **Passo 6**.

Passo 2: **Se** $tabela[ind] \cdot ja\_foi\_unido = 1$ faça o incremento de ind e **Retorne**
Se não siga para o **Passo 3**

Passo 3: **Se** ($tabela[ind] \cdot juntar\_alguem = 1$) **então** decodifique um *flag* de junção e armazeme em $flag\_junta$ e vá para o **Passo 4**.
Se não siga para o **Passo 5**

Passo 3: **Se** ($tabela[ind] \cdot juntar\_dois = 0$) **então**

3.1 **Se** $(flag\_junta = 1)$ **então** decodificamos um índice referente a um bloco junção, armazene este bloco em $\overline{B^j}$, incremente o índice da tabela ind e **Retorne**.

3.1 **Se** $(flag\_junta = 0)$ **então** vá para o **Passo 5**.

Passo 4: **Se** ($tabela[ind] \cdot juntar\_dois = 1$) **então**

4.1 Decodifique um *flag* referente a posição da junção.

4.2 **Se** $(flag\_junta = 1)$ **então** decodificamos um índice referente a um bloco junção, de acordo com sua posição, armazene este bloco em $\overline{B^j}$, incremente o índice da tabela ind e **Retorne**.

4.3 **Se** $(flag\_junta = 0)$ **então** vá para o **Passo 5**.

Passo 5: **Se** ($tabela[ind] \cdot juntar\_alguem = 0$) **então** deve-se ler um índice do bloco, armazena-lo em $\overline{B^j}$, incrementar o índice da tabela ind e **Retorne**.

Passo 6: **Se**$(M > N)$ **entao** faça

$\overline{B^{2j+1}}$ = `DecodificaIndice`(N, $\frac{M}{2}$, $2\,no+1$, $A(n_0)$, $tabela$)

$\overline{B^{2j+2}}$ = `DecodificaIndice`(N, $\frac{M}{2}$, $2\,no+2$, $A(n_0)$, $tabela$)

Se Não faça

$\overline{B^{2j+1}}$ = `DecodificaIndice`($\frac{N}{2}$, M, $2\,no+1$, $A(n_0)$, $tabela$)

$\overline{B^{2j+2}}$ = `DecodificaIndice`($\frac{N}{2}$, M, $2\,no+1$, $A(n_0)$, $tabela$)

Passo 7: Idêntico ao **Passo 8** do procedimento `CodificaIndice`.

Passo 8: Idêntico ao **Passo 9** do procedimento `CodificaIndice`.

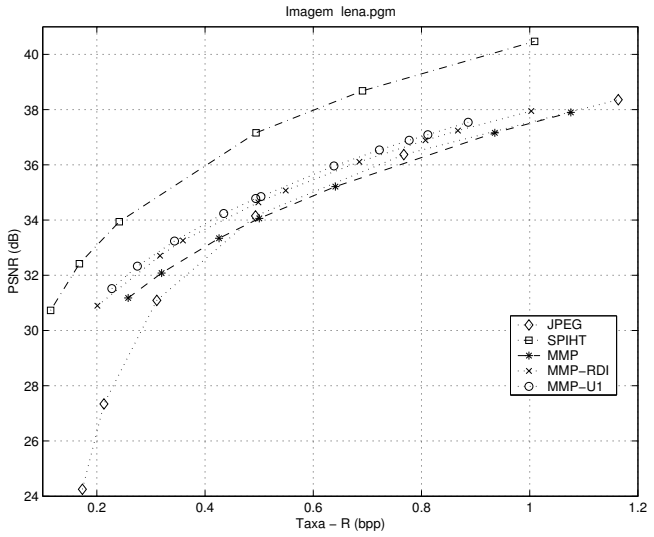
4.2.4 Resultados

Nesta seção iremos apresentar os resultados obtidos pelo MMP-U1, esses são compostos por curvas taxa-distorção e por detalhes das imagens codificadas. A taxa R é dada em bits por pixel (bbp) e a distorção é dada pela $PSNR$ definidas na seção 2.4. Os testes foram feitos com as mesmas imagens usadas pelo seção 3.4, ou seja, temos cinco imagens *pp1205, pp1209, aerial, barbara, lena, f16* e *gold*, todas no formato *pgm*, e com tamanho de 512×512. Para facilitar nossa comparação, em um mesmo gráfico, encontram-se as curvas para o JPEG, SPIHT, MMP e o MMP-RDI. Para nossos testes consideramos que:

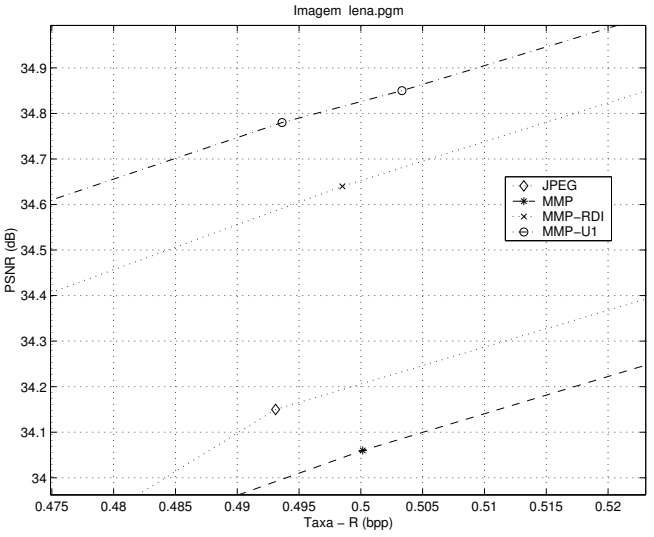
- Temos todas as imagens divididas em blocos de tamanho 16×16 (veja a figura 3.2) onde cada bloco é processado individualmente da esquerda para a direita e de cima para baixo.
- Os subdicionários possuem um tamanho de 150000 blocos.
- A inicialização do dicionário encontra-se descrita na seção 4.2.2 item *B*.
- Temos os modelos estatisticos para os *flags* de posição, junção, segmentação e os índices descritos na tabela 4.5.

4.2.4.1 Curvas taxa-distorção

A figura 4.19, apresenta a curva taxa-distorção para a imagem *lena*. Na figura 4.19(a) temos as curvas relativas. Para esta imagem tivemos um ganho 0,2dB, em relação aos resultados do MMP-RDI, podendo ser visualizado na figura 4.19(b).



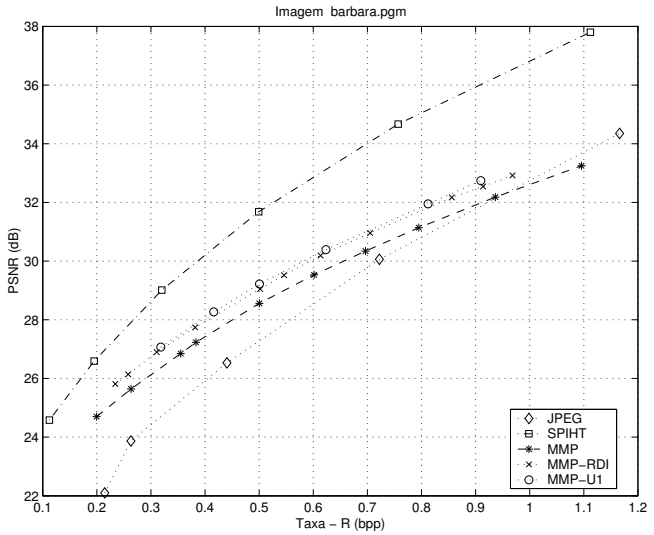
(a) Curva taxa-distorção



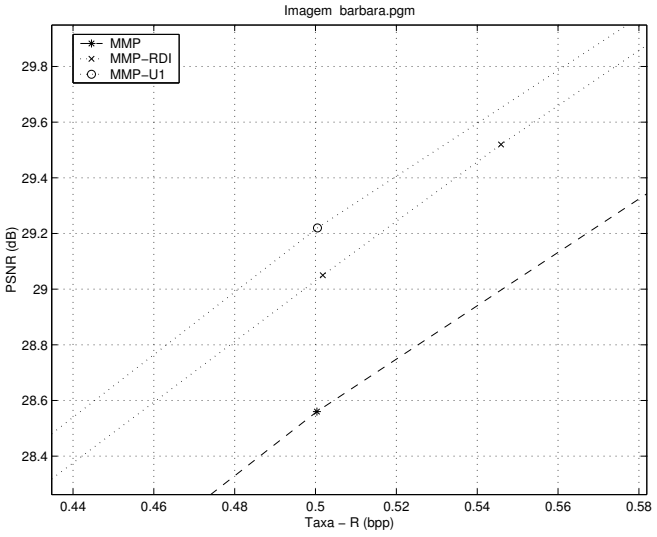
(b) Detalhe

Figura 4.19: Curva taxa-distorção para a Imagem *lena*

A figura 4.20, apresenta a curva taxa-distorção para a imagem *barbara*. As curvas relativas são apresentadas na figura 4.20(a). Para esta imagem também tivemos um ganho de 0,2dB, em relação aos resultados do MMP-RDI, veja a figura 4.20(b).



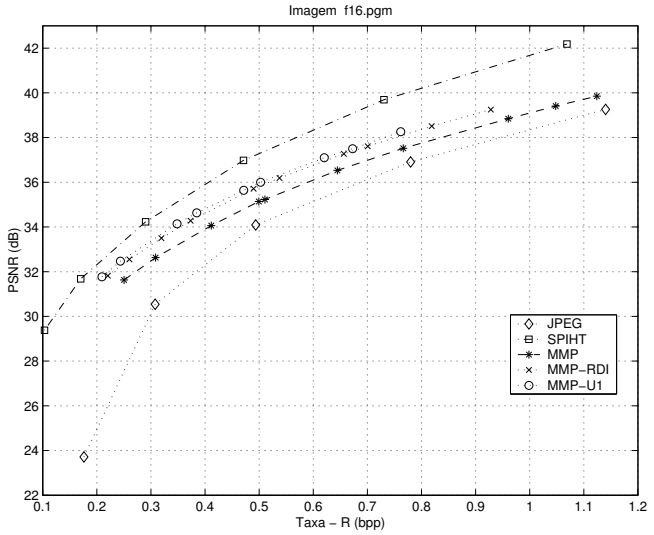
(a) Curva taxa-distorção



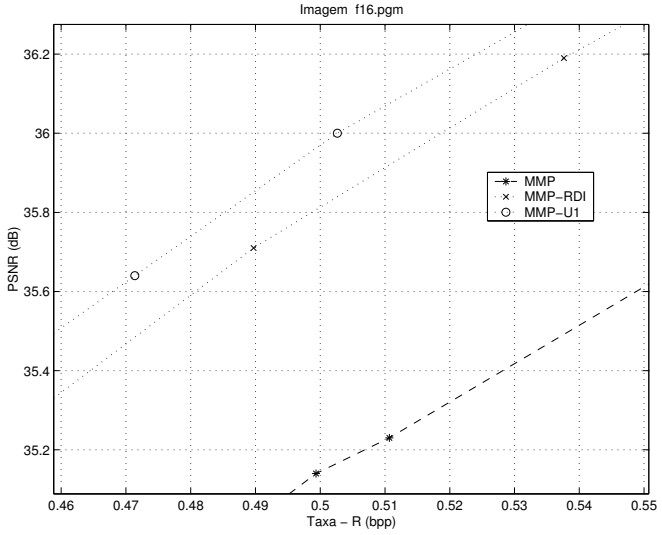
(b) Detalhe

Figura 4.20: Curva taxa-distorção para a Imagem *barbara*

A figura 4.21, apresenta a curva taxa-distorção para a imagem *f16*. No detalhe, percebemos um ganho de aproximadamente 0,2dB, em relação aos resultados do MMP-RDI, veja a figura 4.21(b).



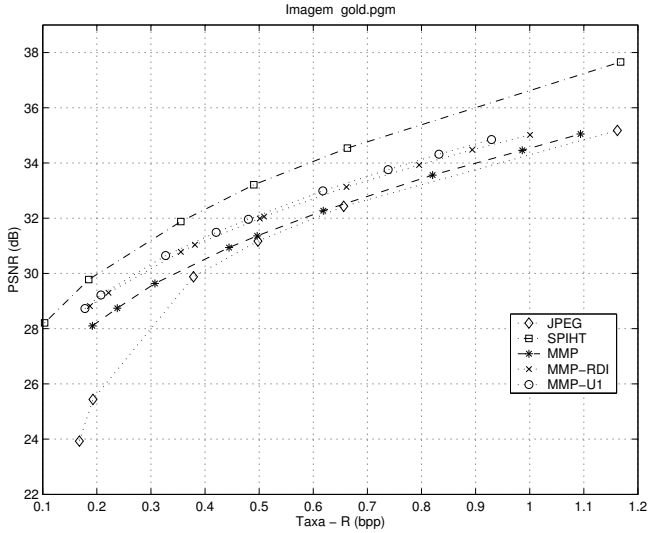
(a) Curva taxa-distorção



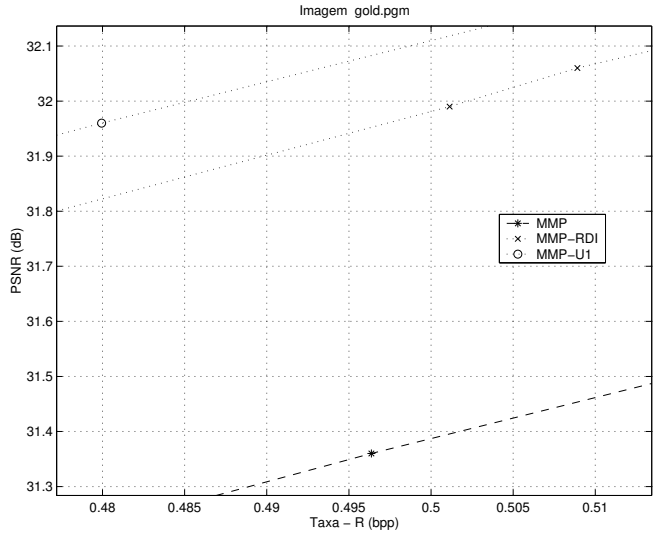
(b) Detalhe

Figura 4.21: Curva taxa-distorção para a Imagem *f16*

A figura 4.22, apresenta a curva taxa-distorção para a imagem *gold*. Essa imagem teve um ganho de 0,15dB, em relação aos resultados do MMP-RDI.



(a) Curva taxa-distorção



(b) Detalhe

Figura 4.22: Curva taxa-distorção para a Imagem *gold*

A seguir temos as curvas para as imagens *aerial* (figura 4.23), *pp1209* (figura 4.25) e *pp1205* (figura 4.24). Na imagem *pp1205* tivemos um ganho negativo de 0,2dB, em 0,5bpp e as imagens *aerial*, *pp1209* mantiveram seus resultados, todos esses em relação aos resultados do MMP-RDI.

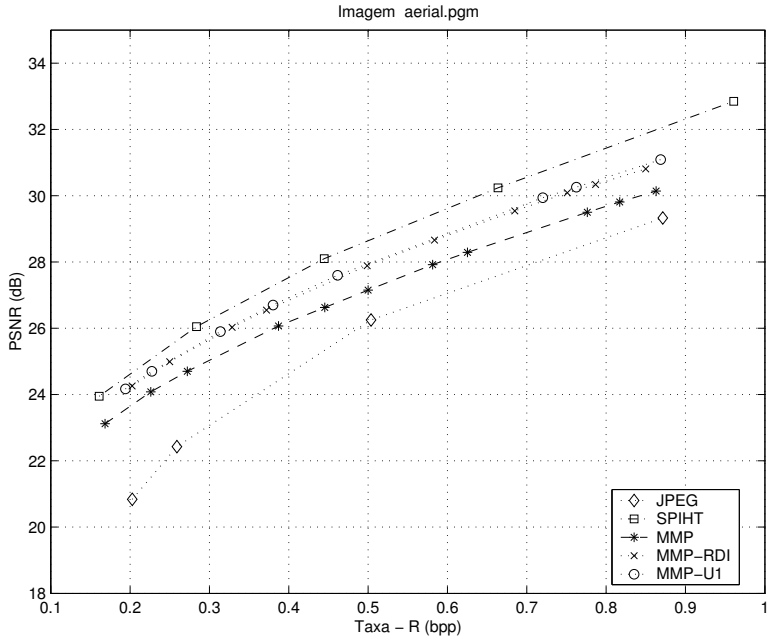


Figura 4.23: Curva taxa-distorção para a Imagem *aerial*

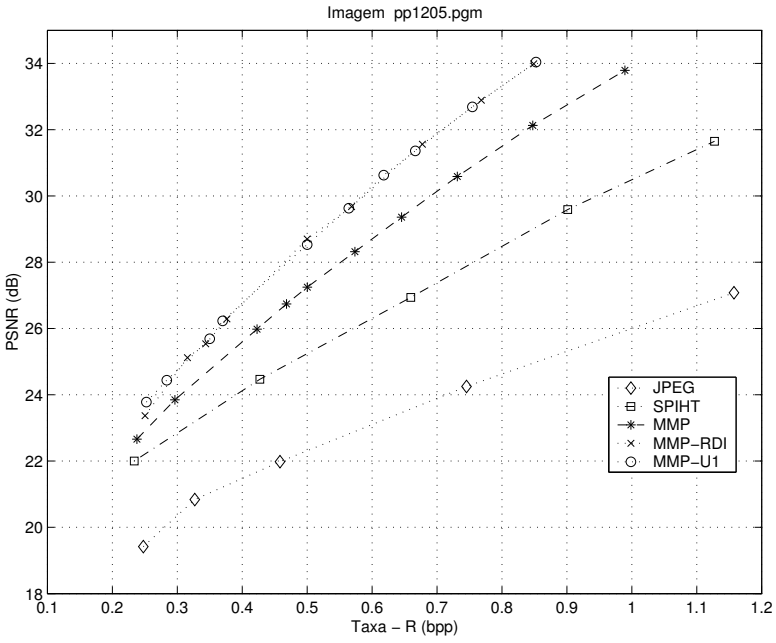


Figura 4.24: Curva taxa-distorção para a Imagem *pp1205*

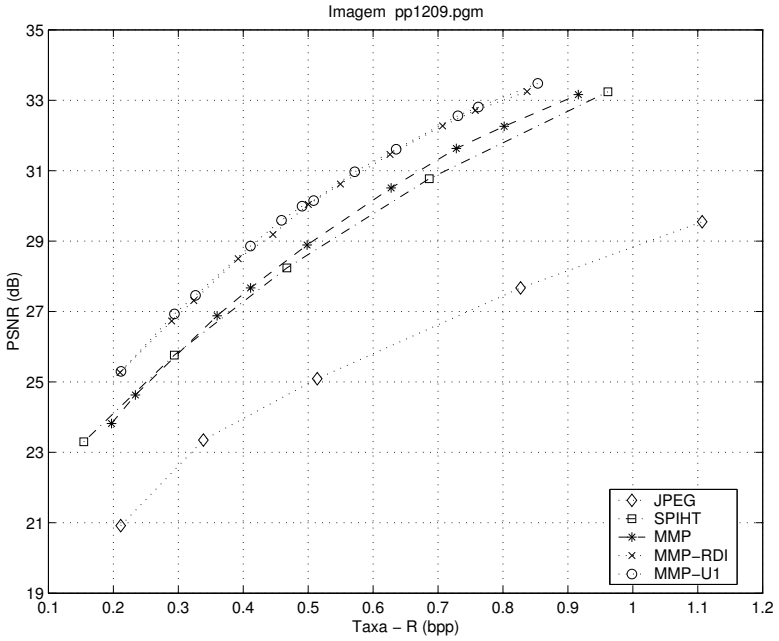
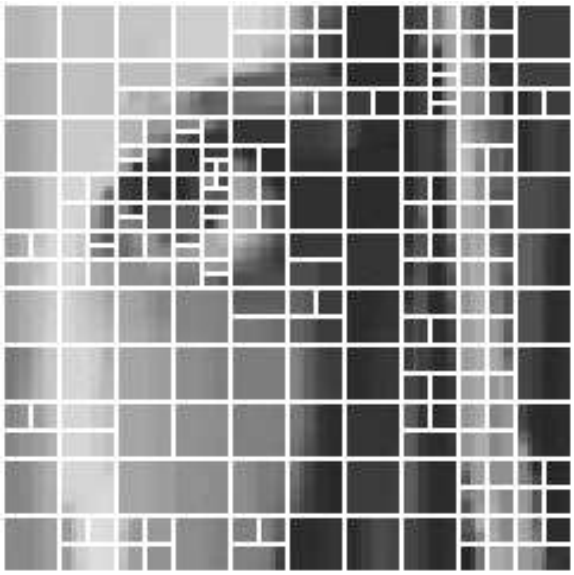


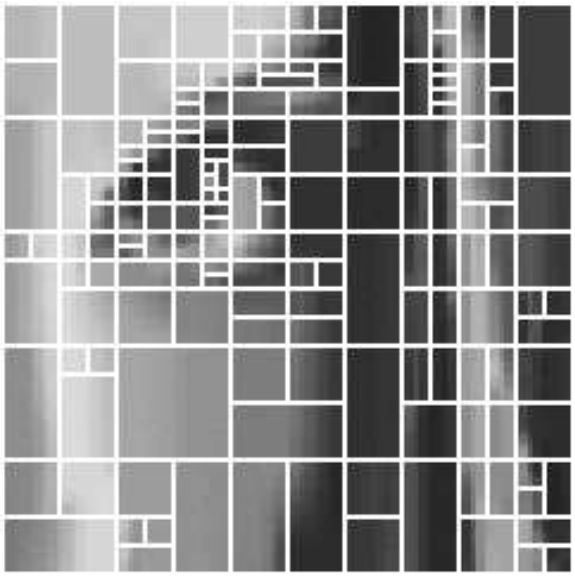
Figura 4.25: Curva taxa-distorção para a Imagem *pp1209*

4.2.4.2 Detalhes das imagens codificadas

Abaixo temos detalhes retirados da imagem *lena*, processada pelo MMP-RDI e pelo MMP-U1. Na figura 4.26 temos a visualização dos limites dos blocos sem a união (figura 4.26(a)) e com a união (figura 4.26(b)).



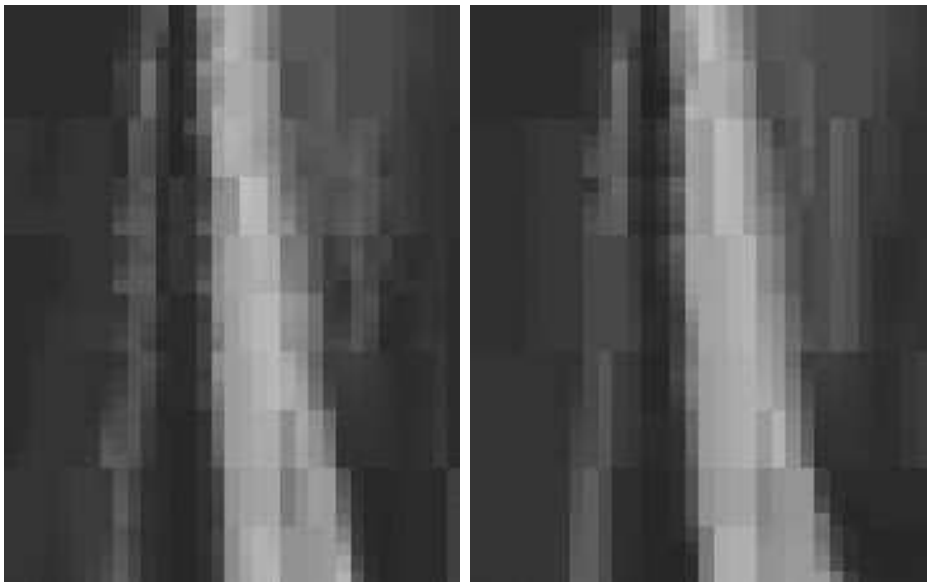
(a) Detalhe MMP-RDI



(b) Detalhe MMP-U1

Figura 4.26: Detalhe para *lena*

Um detalhe bastante importante é a diminuição do efeito indesejado causado pela percepção dos limites dos blocos pois a quantidade desses limites diminuem devido a união de blocos, esse efeito pode ser visto na figura 4.19(b).



(a) MMP-RDI (b) MMP-U1

Figura 4.27: Diminuição da quantidade de limites dos blocos

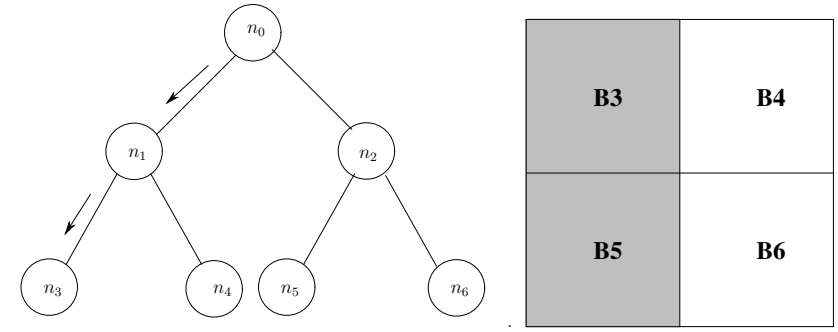
4.3 Comentários

Neste capítulo investigamos a primeira solução para adaptar o MMP a união de blocos. Este primeiro estudo atingiu resultados equivalentes ou um pouco superiores quando comparados ao MMP-RDI porém este método possui um problema que deve ser análisado. O MMP-U1 falha no sentido de disponibilizar valores de custo reais para a ánalise de união, vejamos como isso ocorre.

Como vimos, a união de blocos não obedece o sentido de formação da árvore de segmentação gerada pela função de `OtimizacaoRDI`, sendo assim não é possível obter valores reais de custo pois a avaliação de união sempre será feita com um bloco que ainda não teve suas estatisticas atualizadas pelo simples fato de que este ainda não foi processsado. Vejamos um exemplo.

Neste temos uma árvore otimizada no sentido RD fornecida pela função

`OtimizacaoRDI`, conforme o sentido de formação da árvore o primeiro nó à ser processado é n_3. Para este temos somente uma possível união com n_5. Para realizarmos a avaliação de união precisamos dos custos J_{n3}, J_{n5} e $J_{n3,5}$.



(a) Árvore (b) Bloco associado

Figura 4.28: Primeira avaliação de união

A questão é que J_{n5} não pode ser calculado de um forma precisa pois esse nó ainda não foi processado, ou seja, o custo J_{n5} é estimado e dessa forma a análise de união torna-se aproximada.

Assim, o MMP-U1 não disponibiliza os valores reais dos custos o que compromete a análise de união, portanto seu comportamento não é ótimo.

No próximo capítulo investigaremos um método de união, chamado de MMP-U3, que corrige esse problema, ou seja, não existe mais uma análise de união aproximada, os valores de custos disponibilizados pelo método são reais.

Capítulo 5

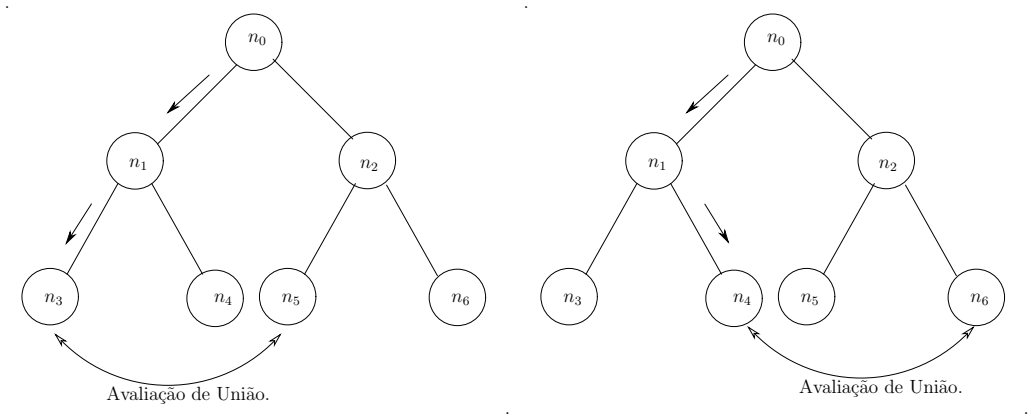
Análise das Estimativas de Custo para União (MMP-U3)

Neste capítulo apresentaremos a solução para contornar o problema do primeiro método de união de blocos (MMP-U1). Conforme comentamos, o MMP-U1 não disponibilizava valores de custos reais e sim uma aproximação. Esse problema foi investigado e sua solução encontra-se no método que chamados de MMP-U3. A seguir temos a descrição introdutória do método, seu algoritmo, os resultados obtidos e a conclusão.

5.1 Introdução

No primeiro estudo do algoritmo de união tínhamos o problema da falha das estatísticas de flags e índices o que ocasionava um cálculo de custo aproximado comprometendo, em parte, a avaliação da união.

Para uma melhor visualização deste problema vamos considerar a árvore, fornecida pela função `OtimizacaoRDI`, abaixo. O primeiro nó processado é n_3 e a única avaliação de união para este é com n_5 (bloco abaixo ou a direita), porém o nó n_5 ainda não foi processado portanto seu custo é aproximado, essa situação foi mostrada na conclusão do capítulo 4. O mesmo fato ocorre quando estamos avaliando a união de n_4 e n_6.

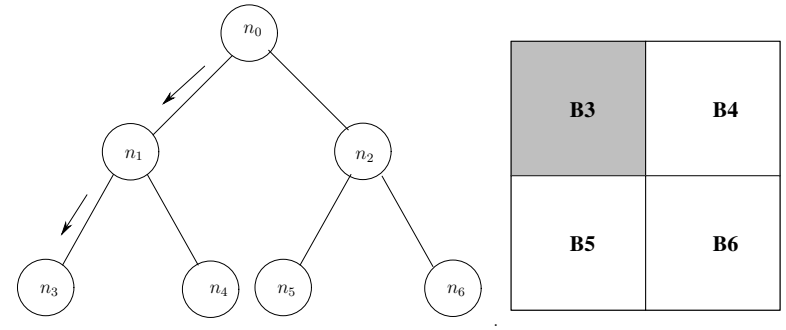


(a) União de n_3 e n_5 (b) União de n_4 e n_6

Figura 5.1: Avaliações de Uniões

O novo método de união propõe uma inversão no sentido de avaliação das uniões, ou seja, as avaliações devem ser feitas com os blocos acima e a esquerda do nó/bloco atual. Dessa forma a avaliação de união aguarda os nós de teste serem processados e somente depois a avaliação é feita.

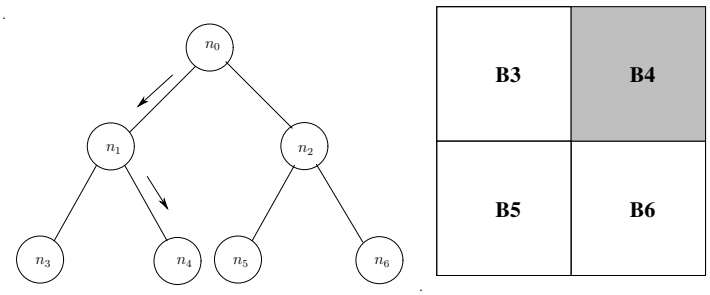
As figuras 5.2, 5.3, 5.4 e 5.5 mostram a evolução do processamento dos nós. O primeiro nó processado é n_3 que não possui nenhuma avaliação a ser feita (bloco acima ou a esquerda) veja 5.2.



(a) Árvore (b) Blocos associados

Figura 5.2: Processamento de n_3

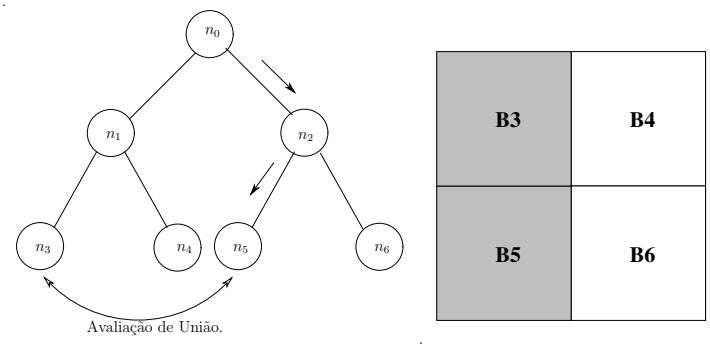
O segundo nó é n_4 e este também não possuí nunhuma potencial união com o de cima ou o da direita. Veja a figura 5.3.



(a) Árvore (b) Blocos associados

Figura 5.3: Processamento de n_4

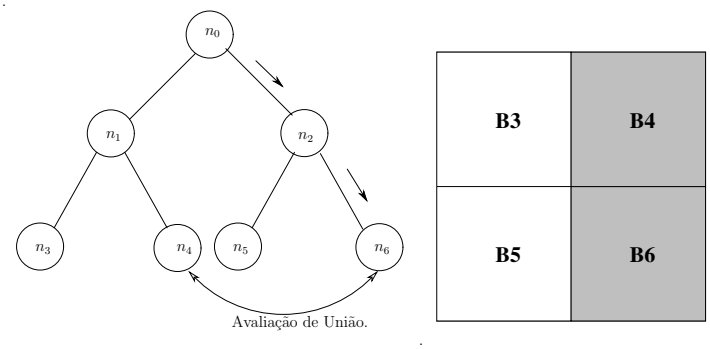
O terceiro nó é n_5. Neste temos a primera avalição de união realizada com n_3. Desta forma, como já processamos n_3 então seu custo não é mais aproximado portanto a avaliação de união deixa de ser uma aproximação, veja a figura 5.4.



(a) Árvore (b) Blocos associados

Figura 5.4: Processamento de n_5

O último nó é n_6. Neste temos a segunda avalição de união feita com n_4. Desta forma, como já processamos n_4 então seu custo não é mais aproximado portanto a avaliação de união, da mesma forma que na primeira avaliação deixa de ser uma aproximação, veja a figura 5.5.



(a) Árvore (b) Blocos associados

Figura 5.5: Processamento de n_6

Isso resolve, em parte, o problema das estatísticas pois em qualquer teste de união todos os nós envolvidos já teriam sido processados no sentido de formação da árvore de segmentação.

Quando se decide pela união dos blocos devemos imediatamente reiniciar o processamento de todos os nós, ou seja, a árvore é reiniciada e começamos a partir de n_0. A união é feita assim que o nó, marcado para união, for processado, sem a avaliação de custo algum pois já decidiu-se pela união.

A reinicialização deve-se ao fato de que os nós entre os blocos unidos devem ter seus custos calculados com as estatísticas atualizadas. Em nosso exemplo, a conseqüência imediata da reinicialização é que o custo de n_4 irá contemplar união $n_{3,5}$ portanto seu custo será real, caso isso não fosse realizado então o custo de n_4 não seria correto pois este nó já teria sido processado e por conseqüência seu custo já teria sido calculado e não contemplaria a união ocorrida.

5.2 Algoritmo MMP-U3

Nesta sessão apresentaremos o algoritmo MMP-U3 cujo objetivo principal é disponibilizar valores de custos reais para que a avaliação de união possa ser realizada sem estimações.

As avaliações de união são invertidas em relação ao MMP-U1, ou seja, ela é realizada com o bloco de cima e/ou o da esquerda, dessa forma, o método *aguarda* a árvore processar o candidato a união e a avaliação só é realizada neste momento. Quando decide-se por uma união devemos imediatamente re-iniciar o processamento para não comprometer as estatísticas. Sendo assim, quando re-iniciamos devemos realizar as união, caso existam, como no MMP-U1 (direita e baixo) pois o bloco já foi marcado para unir.

Da mesma forma que o MMP-U1, é extremamente importante à ordem na qual as funções são chamadas ela é a mesma definida para o algoritmo MMP-U1. As funções `CodificaArvore` e `CodificaIndice` não sofreram nenhuma alteração, o que oferece uma facilidade para migrarmos do MMP-U1 para o MMP-U3 em contra partida as funções `GeraTabela` e `OtimizacaoJuncao` crescem em complexidade. Os mesmos modelos de freqüência são utilizados, veja a tabela 4.4.

A decodificação não sofre nenhuma alteração.

5.2.1 Função `GeraTabela`

Esta função é responsável pela geração da tabela de união. A tabela de união possui campos adicionais em relação a tabela descrita em 4.5 esses referem-se às uniões com o bloco de cima e/ou com o da esquerda. Podendo ser facilmente extendida, seus campos adicionais são idênticos aos campos de número 2, 6, 7, 9, 10, 1, 12 da tabela 4.5. Neste algoritmo esses campos aparecem com a identificação "$\_u3$".

A função `GeraTabela` é composta por três funções, a `GeraTabela1`, `GeraTabela2` e `GeraTabela3`. As duas primeiras funções estão descritas em 4.2.3.2. A função `GeraTabela3`, basicamente, avalia potências uniões de blocos com o de cima ou o da esquerda. Seu algoritmo encontra-se abaixo.

Procedimento `GeraTabela3`(N, M, no, $A(n_0)$, $tabela$);

Passo 1: **Se** $(A(n_{2\,no+1}) = A(n_{2\,no+2}) = 0)$ **ou** (N=M=1) **então** segue para o **Passo 2**.

Se não vai para o **Passo 5**.

Passo 2: **Se**$(no\,\%\,2 = 0)$, ou seja, o nó é par **então**

Se $M > N$ **então**

Observa se existe possibilidade de união com o bloco da esquerda

Se existir possibilidade **então** faça

Guarde o índice da tabela referente ao bloco que uniu à esquerda em m.

$tabela[ind] \cdot (Nj1\_u3, Mj1\_u3) = (tabela[ind].N,\ tabela[ind].M + tabela[m].M)$

$tabela[ind] \cdot juntar\_alguem\_u3 = 1$

$tabela[ind] \cdot juntar\_dois\_u3 = tabela[ind] \cdot flag\_posicao\_u3 = 0$

$tabela[ind] \cdot juncao1\_u3 = m$

Se não

$tabela[ind].juntar\_alguem\_u3 = tabela[ind].juntar\_dois\_u3 = 0$

Se não

Observa se existe possibilidade de união com o bloco de cima

Se existir possibilidade **então** faça

Guarde o índice da tabela referente ao bloco que uniu com o de baixo em m.

$tabela[ind].(Nj1\_u3, Mj1\_u3) = (tabela[ind].N + tabela[m].N,\ tabela[ind].M)$

$tabela[ind].juntar\_alguem\_u3 = 1$

$tabela[ind].juntar\_dois\_u3 = 0$

$tabela[ind].flag\_posicao\_u3 = 1$

$tabela[ind].juncao1\_u3 = m$

Se não

$tabela[ind].juntar\_alguem\_u3 = tabela[ind].juntar\_dois\_u3 = 0$

Passo 3: **Se**$(no\,\%\,2 = 1)$, ou seja, o nó é ímpar **então**

Verifica se existe possibilidade de união tanto com o da esquerda

quanto com o de cima.

Se ambas existirem **então**

$tabela[ind].juntar\_alguem\_u3 = tabela[ind].juntar\_dois\_u3 = 1$

Se existir apenas uma **então**

$tabela[ind].juntar\_alguem\_u3 = 1$

$tabela[ind].juntar\_dois\_u3 = 0$

Seta o campo $tabela[ind].flag\_posicao\_u3$ de acordo com

o tipo (com o da esquerda ou o de cima)

Se não existir nenhuma **então**

$tabela[ind].juntar\_alguem\_u3 = tabela[ind].juntar\_dois\_u3 = 0$

Passo 4: incremente o índice da tabela *ind* e **Retorna**.

Passo 5: **Se**$(M > N)$ **entao** chame a função `GeraTabela3` na ordem:

`GeraTabela3`(N, $\frac{M}{2}$, $2\,no + 1$, $A(n_0)$)

`GeraTabela3`(N, $\frac{M}{2}$, $2\,no + 2$, $A(n_0)$)

Se Não chame a função `GeraTabela3` na ordem:

`GeraTabela3`($\frac{N}{2}$, M, $2\,no + 1$, $A(n_0)$)

`GeraTabela3`($\frac{N}{2}$, M, $2\,no + 2$, $A(n_0)$)

5.2.2 Função `OtimizacaoJuncao`

Esta função realiza a avaliação das uniões a cima e esquerda e decide se tais uniões devem ser codificadas ou não. É importante observar a chamada da função pois devemos controlar as re-inicializações.

Chamada da função

$otimizacao\_break = 1$

enquando faça

`OtimizacaoJuncao`(16, 16, 0, $A(n_0)$, $tabela$)

Se ($otimizacao\_break = 1$) **então**

break

Se não faça

Zerar os contadores espelho e todos os índices

do campo $ja\_foi\_unido\_u3$.

$otimizacao\_break = 1$

Procedimento $\overline{B^j}$ = `OtimizacaoJuncao`(N, M, B^j, no, $A(n_0)$, $tabela$);

Passo 1: **Se** ($A(n_{2\,no+1}) = A(n_{2\,no+2}) = 0$) **ou** (N=M=1) **então** vai para o **Passo 2** caso contrário vai para o **Passo 10**

Passo 2: **Se** ($tabela[ind] \cdot ja\_foi\_unido\_u3 = 1$) **então** faça o incremento de ind e **Retorne**.

Passo 3: **Se** ($tabela[ind] \cdot flag\_junta = 1$)

3.1 Localizar, na tabela de união, o bloco marcado para a união que foi escolhido antes da reinicialização, guarde seu índice da tabela em tmp e armazenar em i o índice do dicionário que melhor aproximou o bloco união, chamdo de s_i.

3.2 Faça $otimizacao\_break = 1$ e sete os seguintes campos das tabelas:
$tabela[ind] \cdot flag\_junta\_u3 = tabela[tmp] \cdot flag\_junta\_u3 = 1$
$tabela[ind] \cdot ja\_foi\_unido\_u3 = tabela[tmp] \cdot ja\_foi\_unido\_u3 = 1$
Sete $tabela[ind] \cdot flag\_posicao\_u3$ de acordo com a posição.

3.3 Faça $\overline{B^j} = s_i$, incremente o flag de junção $f\_fs\_o(1)$ e o índice i como:
Se (A aproximação s_i foi feita por D) **então** incremente fio.
Se (A aproximação s_i foi feita por D_R) **então** incremente fir.
Incremente o índice da tabela ind e guarde o bloco aproximado em $\overline{B^j}$.

3.4 Vai para o **Passo 9**.

Passo 4: **Se** $tabela[ind] \cdot juntar\_alguem\_u3 = 1$ **então** siga para o **Passo 5**.
Se não Siga para o **Passo 7**.

Passo 5: **Se** ($tabela[ind] \cdot juntar\_dois\_u3 = 0$) **então** siga para *5.1*
Se não siga para o **Passo 6**

5.1 Recupere o custo da união com o qual a folha atual se junta, guarde em CJ.

5.2 Recupere o índice da tabela referente ao bloco com o qual o nó atual se une guarde este em tmp e o seu custo guarde em C_{nb}.

5.3 Calcule o custo para o bloco atual, ou seja, $C_{na} = D_{n_j} + \lambda RI_{n_j}$. Guarde o índice da aproximação do dicionário (D ou D_R) em i, e faça $\overline{Ba^j}$ igual a aproximação.

5.4 Calcule a taxa R_{J0} para o flag de junção igual a zero que será usado para para informar que o nó atual não sofre união.

5.5 Faça a avaliação de união, ou seja:
Se $\{CJ+\} \leq \{C_{na} + \lambda R_{J0} + C_{nb}\}$ **então** vai para *5.6*
Se não siga para o Passo 7

5.6 Faça:
Os blocos devem ser marcados como união (Setando-se os campos da tabela).
$otimizacao\_break = 0$ e **Retorna** /* re-inicializa */

Passo 6: **Se** ($tabela[ind] \cdot juntar\_dois\_u3 = 1$) **então**

6.1 Calcule o custo para o bloco atual, ou seja, $C_{na} = D_{n_j} + \lambda RI_{n_j}$. Guarde o índice da aproximação do dicionário (D ou D_R) em i, e faça $\overline{Ba^j}$ igual a aproximação. Calcule a taxa R_{J0}.

6.2 Recupere os dados da primeira união, ou seja, o custo da união (CJ), o custo do bloco que sofreu união (C_{nb}) e o índice da tabela referente ao bloco que uniu, $tmp1$.

6.3 Realize a primeira avaliação de união, ou seja:
Se $\{CJ\} \leq \{C_{na} + \lambda R_{J0} + C_{nb}\}$ **então** $u_1 = 1$
Se não $u_1 = 0$
Calcule a redução $RED1 = (C_{na} + C_{nb}) - CJ$

6.4 Recupere os dados da segunda união, ou seja, o custo da união (CJ), o custo do bloco que sofreu união (C_{nb}) e o índice da tabela referente ao bloco que uniu, $tmp2$.

6.5 Realize a segunda avaliação de união, ou seja:
Se $\{CJ\} \leq \{C_{na} + \lambda R_{J0} + C_{nb}\}$ **então** $u_2 = 1$

Se não $u_2 = 0$

Calcule a redução $RED2 = (C_{na} + C_{nb}) - CJ$

6.6 **Se** ($u_1 = 0$ **e** $u_2 = 0$) **então**

Vá para o **Passo 7**.

Se ($u_1 = 1$ **e** $u_2 = 0$) **ou** ($u_1 = 0$ **e** $u_2 = 1$) **então**

Os blocos da avaliação que vale 1 devem ser marcados

como união e **Retorne**.

Se ($u_1 = 1$ **e** $u_2 = 1$) **então**

Se $RED1 > RED2$ **então** Os blocos da avaliação u_1 devem ser

marcados como união e **Retorne**.

Se não Os blocos da avaliação u_2 devem ser

marcados como união e **Retorne**.

Passo 7: Calcula o custo do bloco atual e realize a contagem dos índices espelho.

Passo 8: Calcula o custo das junções a direita e a baixo.

Passo 9: incrementa o índice da tabela *ind* e **Retorne**.

Passo 10: **Se** ($M > N$) **então** chame a função `OtimizacaoJuncao` na ordem:

$\overline{B^{2j+1}}$ = `OtimizacaoJuncao`(N, $\frac{M}{2}$, $2\,no + 1$, $A(n_0)$, *tabela*)

Se($otimizacao\_break = 0$) **Retorne**

$\overline{B^{2j+2}}$ = `OtimizacaoJuncao`(N, $\frac{M}{2}$, $2\,no + 2$, $A(n_0)$, *tabela*)

Se($otimizacao\_break = 0$) **Retorne**

Se Não chame na ordem:

$\overline{B^{2j+1}}$ = `OtimizacaoJuncao`($\frac{N}{2}$, M, $2\,no + 1$, $A(n_0)$, *tabela*)

Se($otimizacao\_break = 0$) **Retorne**

$\overline{B^{2j+2}}$ = `OtimizacaoJuncao`($\frac{N}{2}$, M, $2\,no + 2$, $A(n_0)$, *tabela*)

Se($otimizacao\_break = 0$) **Retorne**

Passo 11: Atualize o dicionário D_R da seguinte forma:

Para $n = (0, 1, \ldots, w - 1)$ **faça**

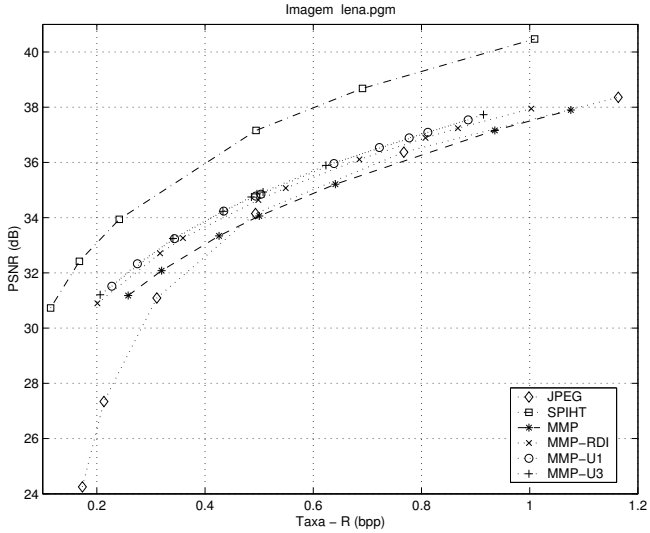
$N_x = 2^{\lfloor 0.5(n+1) \rfloor}$

$$M_x = 2^{\lfloor 0.5n \rfloor}$$
$$S_{N_x,\,M_x} = \{S_{N_x,\,M_x}\} \cup \left\{ T^{N,M}_{N_x,M_x} \left[\overline{B^j} \right] \right\}$$

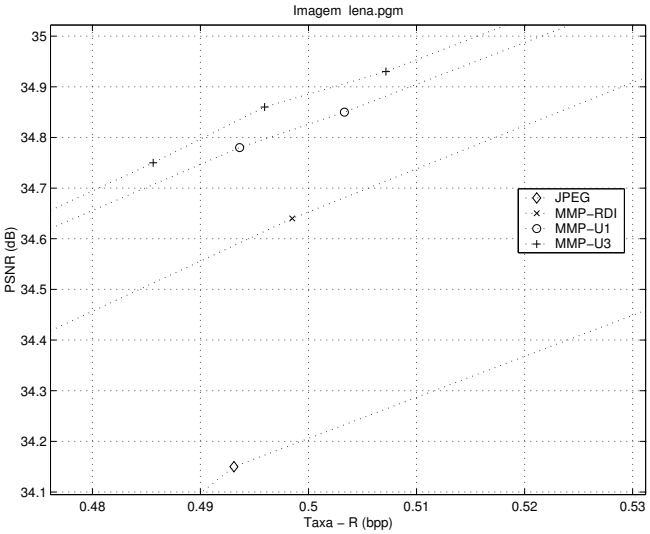
5.3 Resultados

A seguir temos os resultados para toda as imagens de teste, ou seja, *aerial*, *barbara*, *lena*, *f16*, *gold pp1205* e *pp1209* com suas respectivas curvas taxa-distorção. As mesmas considerações definidas na sessão 4.2.4 são usadas aqui.

A figura 5.6(a), apresenta a curva taxa-distorção para a imagem *lena*. Conforme podemos ver em detalhe na figura 5.6(b), os resultados ficaram com um ganho de 0,1dB em relação aos resultados do primeiro método de união, MMP-U1.



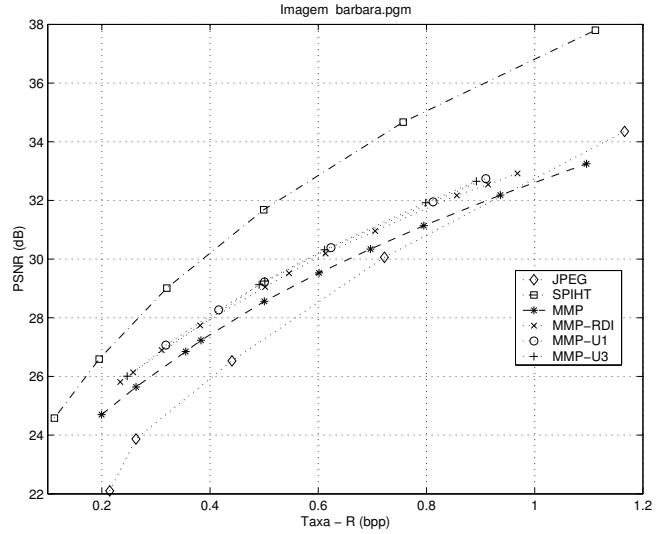
(a) Curva taxa-distorção



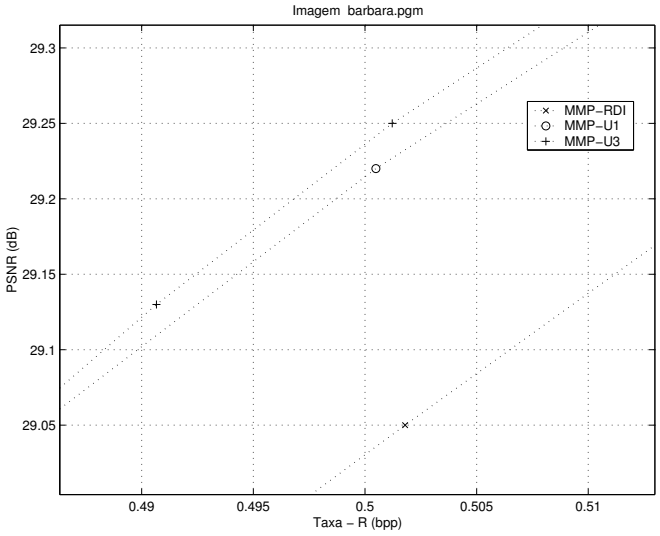
(b) Detalhe

Figura 5.6: Curva taxa-distorção para a Imagem *lena*

A figura 5.7(a), apresenta a curva taxa-distorção para a imagem *barbara*. Nesta imagem, conforme podemos visualizar no detalhe da figura 5.7(b), os resultados ficaram com um ganho de 0,05dB em relação aos resultados do MMP-U1.



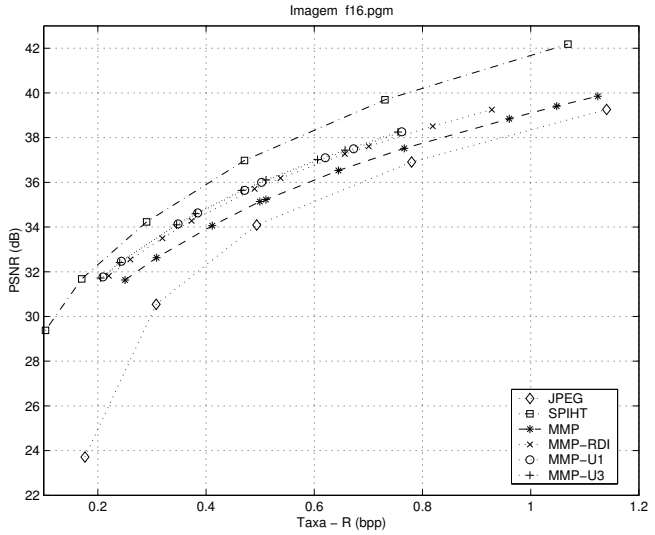
(a) Curva taxa-distorção



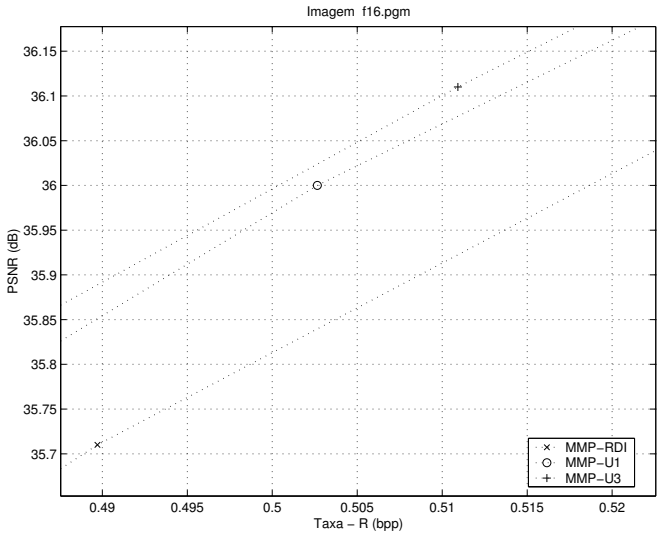
(b) Detalhe

Figura 5.7: Curva taxa-distorção para a Imagem *barbara*

A figura 5.8(a), apresenta a curva taxa-distorção para a imagem *f16*. Nesta, podemos perceber pelo detalhe mostrado na figura 5.8(b) que seus resultados foram aproximadamente iguais aos do MMP-U1.



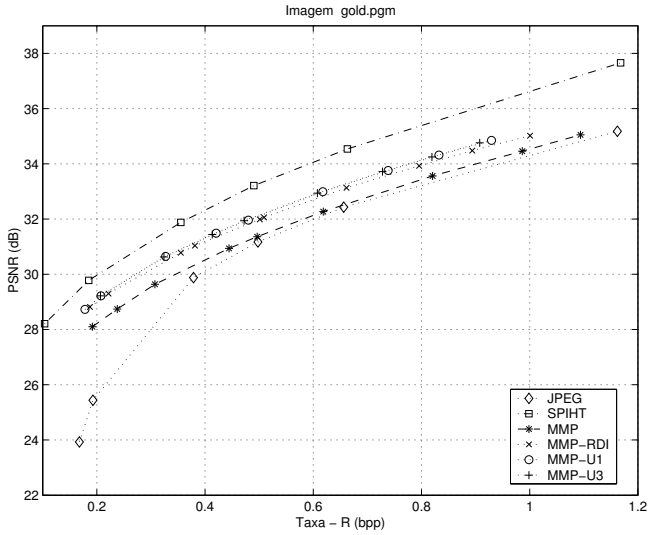
(a) Curva taxa-distorção



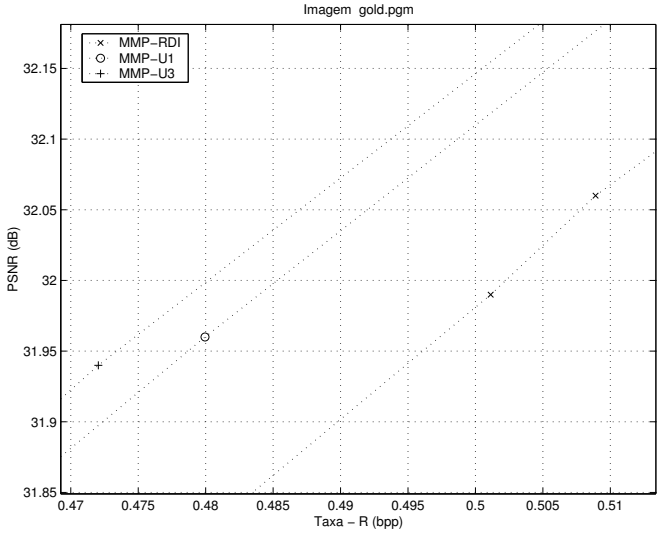
(b) Detalhe

Figura 5.8: Curva taxa-distorção para a Imagem *f16*

A figura 5.9(a), apresenta a curva taxa-distorção para a imagem *gold*. De forma análoga à imagem *f16* seus resultados permaneceram praticamente iguais, podendo ser visualizado na figura 5.9(b).



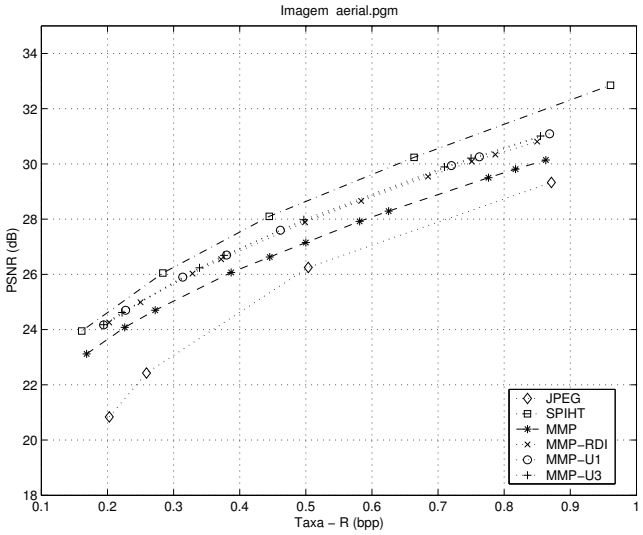
(a) Curva taxa-distorção



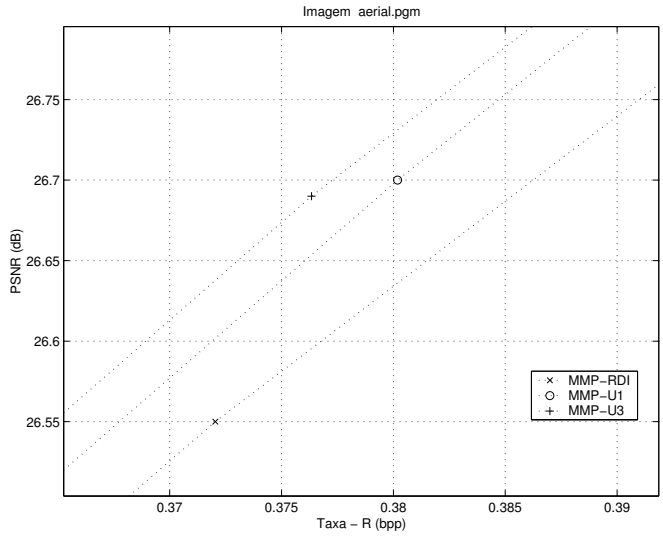
(b) Detalhe

Figura 5.9: Curva taxa-distorção para a Imagem *gold*

A figura 5.10(a), apresenta a curva taxa-distorção para a imagem *aerial*. Podemos verificar pelo detalhe apresentado na figura 5.10(b) que os resultados foram aproximadamente iguais.



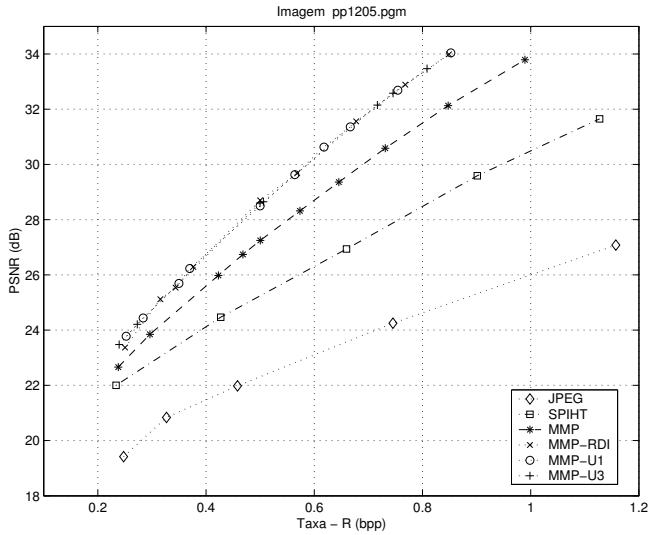
(a) Curva taxa-distorção



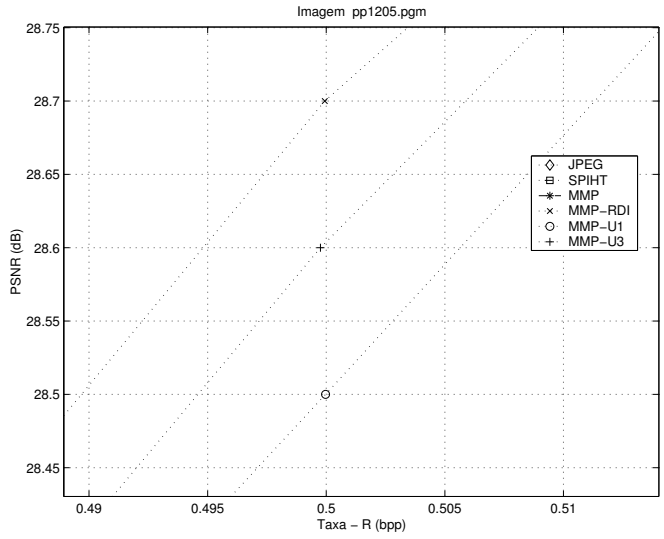
(b) Detalhe

Figura 5.10: Curva taxa-distorção para a Imagem *aerial*

A figura 5.11(a), apresenta a curva taxa-distorção para a imagem *pp1205*. Nesta imagem tivemos um ganho de 0,1dB em relação aos resultados do MMP-U1. Podemos verificar tal fato através da curva em detalhe apresentada na figura 5.11(b)



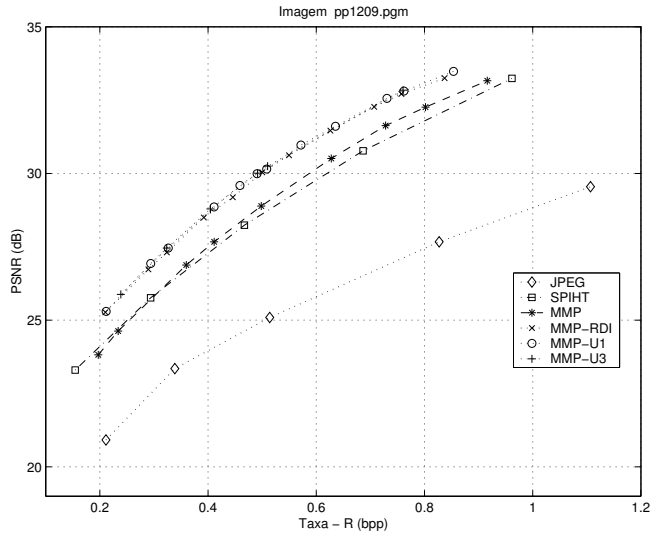
(a) Curva taxa-distorção



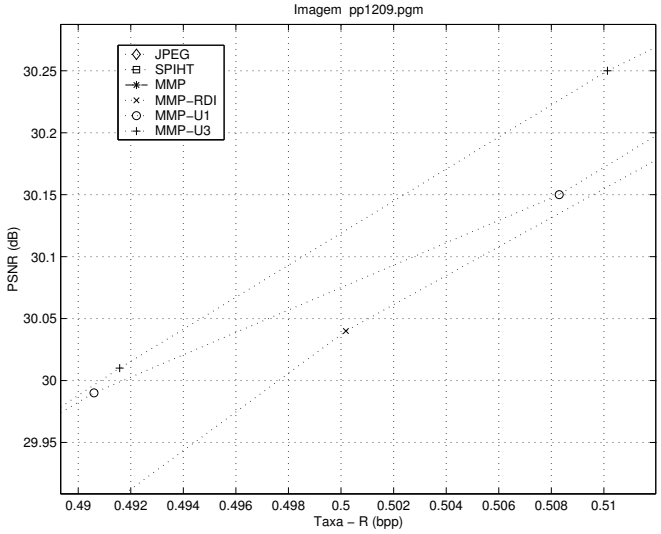
(b) Detalhe

Figura 5.11: Curva taxa-distorção para a Imagem *pp1205*

A figura 5.12(a), apresenta a curva taxa-distorção para a imagem *pp1209*. Nesta imagem os resultados praticamente foram os mesmos encontrados para o MMP-U1.



(a) Curva taxa-distorção



(b) Detalhe

Figura 5.12: Curva taxa-distorção para a Imagem *pp1209*

5.4 Comentários

A análise das estimativas de custo corrige a diferença obtida pela imagem *pp1205* em relação ao MMP-RDI, contudo os resultados para imagens suaves ficam com ganho de, no máximo 0,15dB, não sendo satisfatórios.

Esse problema ocorre divido a falta de padrões nos subdicionários com dimensões de blocos unidos, isso acontece porque a atualização do dicionário para o método de união é a mesma que a do MMP original, ou seja, o dicionário é atualizado com a concatenação dos blocos previamente codificados. Obviamente quando uma união ocorre o dicionário não é atualizado, isto faz com que os dicionário nas escalas correspondentes à uniões sejam apenas versões escaladas dos demais dicionários. Foi observado que, desta forma, os dicionários não aprendem padrões suficientes para codificar os blocos unidos de forma eficiente.

Nó próximo capítulo será investigado um método para aumentar a população destes dicionários chamado de *atualização reversa*.

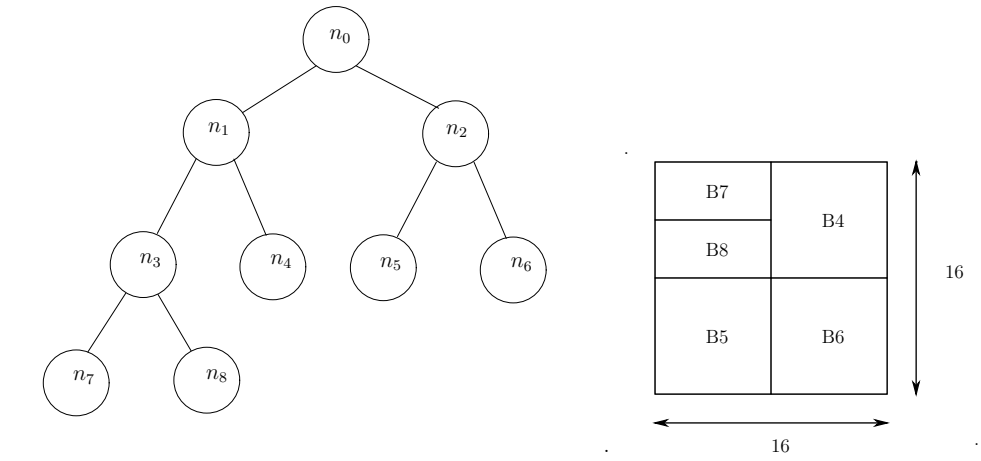
Capítulo 6

Atualização do Dicionário em um Cenário de União de Blocos (Atualização Reversa)

Neste capítulo iremos desenvolver um estudo sobre o efeito do método de atualização do dicionário do MMP para o caso em que temos uniões de blocos. Verificaremos que com esse estudo e a análise de estimativa de custo, desenvolvida no capítulo 5 temos um aumento de desempenho para o método de união de blocos. A seguir iremos apresentar a motivação para mudar o método de atualização dicionário do MMP e os resultados obtidos.

6.1 Motivação

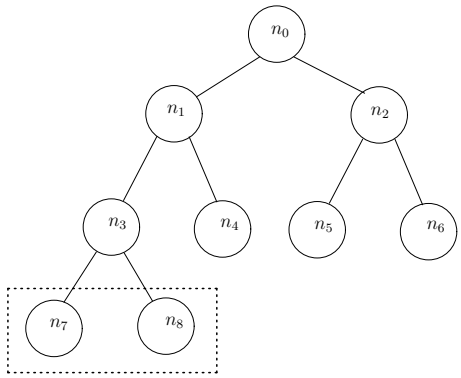
Os métodos de união descritos no capítulo 4,5 possuem uma forma de atualização do dicionário idêntica ao MMP original, ou seja, o dicionário é atualizado com a concatenação dos blocos previamente codificados. Como exemplo, considere uma árvore e seu bloco associado da figura 6.1 da seguinte forma:



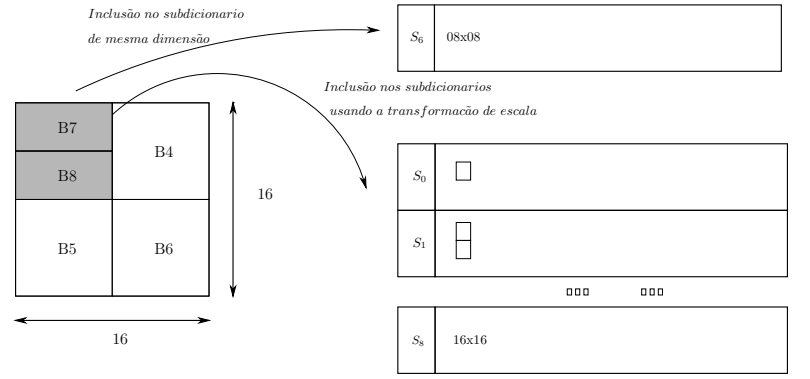
(a) Árvore (b) Bloco associado

Figura 6.1: Árvore de segmentação

Neste exemplo, a primeira atualização do dicionário ocorre quando os nós n_7 e n_8 são codificados, ou seja, concatenamos esses nós e efetuamos a atualização do subdicionário de mesma dimensão da concatenação ($S_6 = (8, 8)$) e dos outros subdicionários aplicando-se a transformação de escala. Na figura 6.2(a) indicamos a posição da atualização em relação a árvore de segmentação e na figura 6.2(b) podemos perceber quais são os subdicionário que recebem um bloco originado da transformação de escala.



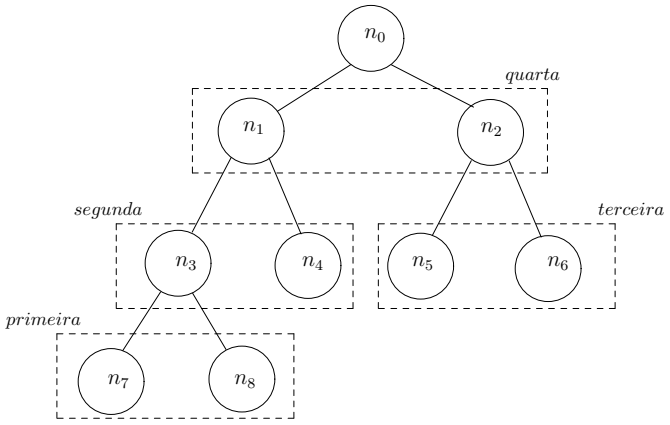
(a) Árvore



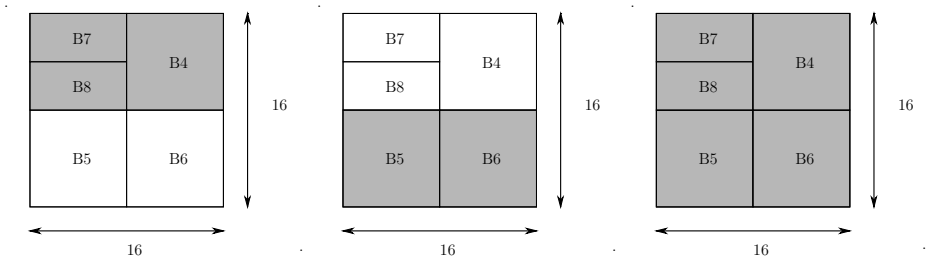
(b) Atualização

Figura 6.2: Primeira atualização do dicionário

A segunda ocorre com a concatenação dos nós n_7,n_8 e n_4 (figura 6.3(b)). A próxima ocorre com n_5 e n_6 (figura 6.3(c)). A última atualização é realizada com a concatenação de n_7,n_8,n_4,n_5 e n_6 (figura 6.3(d)). Todo esse processo pode ser acompanhado, em sequência, pela figura 6.3.



(a) Visualização na árvore



(b) Segunda (c) Terceira (d) Quarta

Figura 6.3: Atualizacões restantes

Porém, quando temos uma união, a atualização é realizada com *parte* do bloco unido e o subdicionário de mesma dimensão da união só recebe blocos originados através da transformação de escala. Para visualisarmos melhor essa situação vamos considerar um exemplo que possui a mesma árvore de segmentação onde $n_{8,5}$ foi marcado para união. Neste caso, a codificação tem a seguinte ordem, n_7, a união $n_{8,5}$, n_4 e n_6, ou seja, a árvore fica como na figura 6.4.

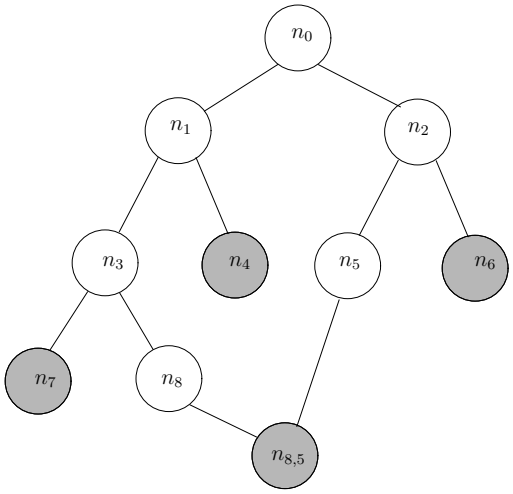
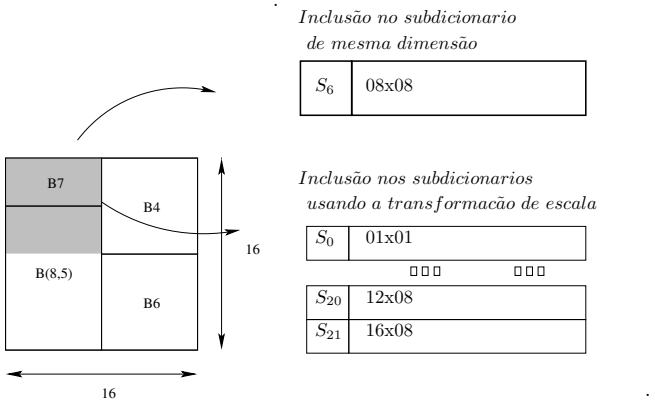


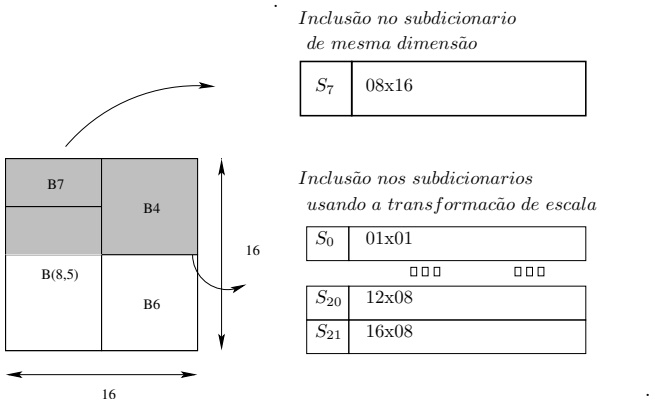
Figura 6.4: Ordem de codificação

Vamos observar como o subdicionário com dimensão $(12, 8)$, ou seja, S_{20}, se comporta quando as atualizações de forma idêntica a figura 6.3(a) ocorrem. A primeira atualização ocorre com a concatenação de n_7 e parte do bloco união $n_{8,5}$ (marcada em cinza na figura 6.5(a)). Na segunda temos a concatenação de n_7 parte de $n_{8,5}$ e n_4. Nessas duas atualizações temos o subdicionário S_{20} recebendo somente blocos originados da transformação de escala (veja as figuras 6.5(b) e 6.6(b)).



(a) Bloco (b) Dicionário

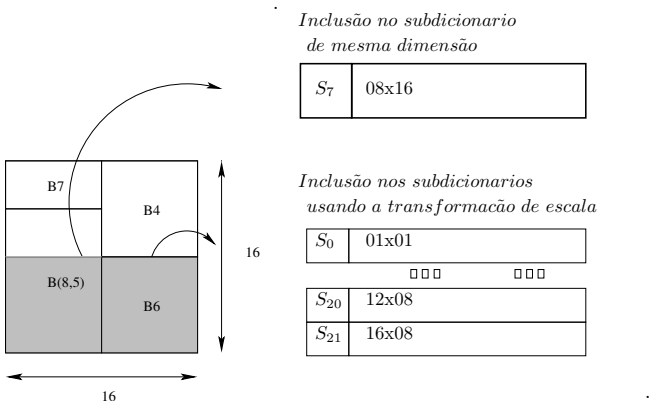
Figura 6.5: Comportamento de S_{20} para a primeira atualização



(a) Bloco (b) Dicionário

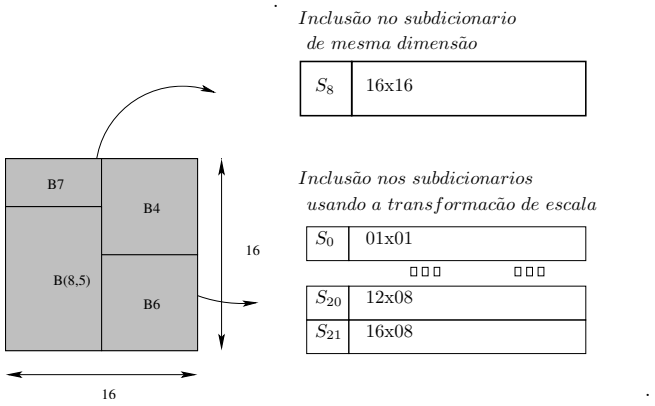
Figura 6.6: Comportamento de S_{20} para a segunda atualização

Na terceira e quarta atualização o comportamento é análogo e em ambas temos que o subdicionário S_{20} também só recebe blocos originados da transformação de escala (veja as figuras 6.7(b) e 6.8(b)).



(a) Bloco (b) Dicionário

Figura 6.7: Comportamento de S_{20} para a terceira atualização



(a) Bloco (b) Dicionário

Figura 6.8: Comportamento de S_{20} para a quarta atualização

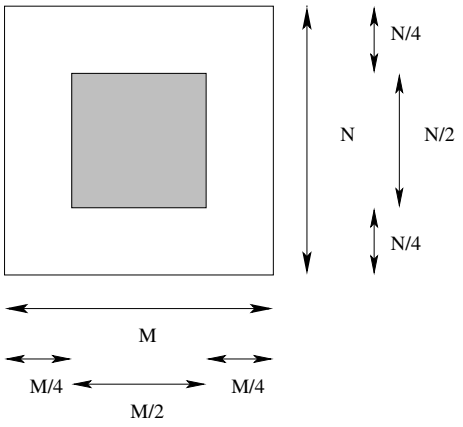
Este fato *compromete a codificação* de blocos pertencentes os subdicionários criados para os blocos de união pois, blocos originados da transformação de escala tem baixa probabilidade de serem usados. Uma outra consequência indesejada que percebemos foi o alto custo que os blocos pertencentes aos subdicionários de união possuem. Em 78% dos casos, os blocos com custo alto são originados de subdicionários de união. A seguir iremos propor uma mudança na forma de atualização do dicionário para diminuir o efeito desse problema.

6.2 Proposta de atualização para união de blocos

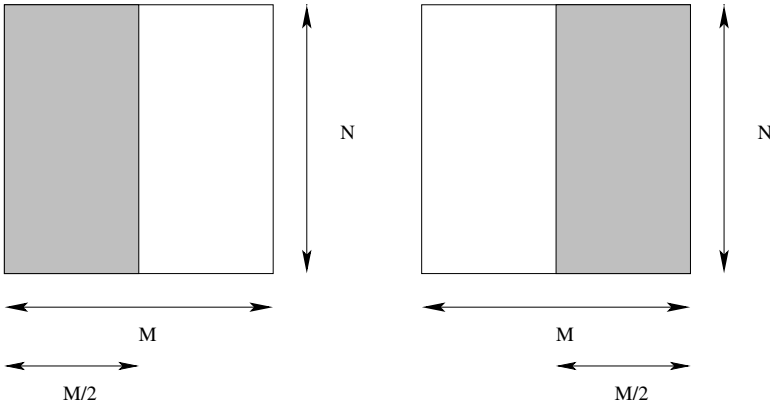
Nossa proposta concentra-se em realizar novas atualizações, no método MMP-U3, além da atualização atual, de tal forma que seja possível deslocar o bloco atualizado de modo a blocos *originais* (não transformados) serem inseridos nos dicionários. Fizemos várias tentativas de deslocamentos e determinamos que a melhor forma é realizar 5 atualizações extras onde os deslocamentos podem ser dados pelas figuras abaixo. Os deslocamentos são realizados dentro do bloco, já concatenado, que irá atualizar o dicionário. Dessa forma o método MMP-U3 com as atualizações reversas passa a chamar-se MMP-U3-AREV. Todas as figuras descrevem como se comporta o deslocamento considerando-se um bloco, já concatenado, de tamanho (N, M). É importante comentar que os dicionários crescem

de forma expressiva, porém tal crescimento somente acarreta um tempo maior de processamento para as imagens.

Abaixo temos os deslocamentos verticais e o deslocamento central indicados nas figuras 6.9(a), 6.9(b) e 6.9(c).



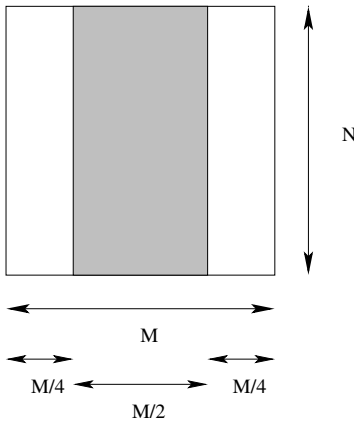
(a) Primeira



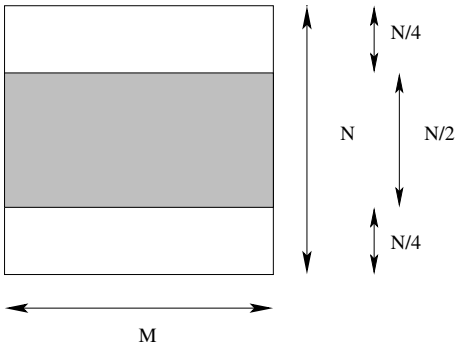
(b) Segunda (c) Terceira

Figura 6.9: Atualizações extras

A seguir temos os deslocamentos horizontais indicados nas figuras 6.10(a) e 6.10(b).



(a) Quarta



(b) Quinta

Figura 6.10: Atualizações extras

6.3 Resultados

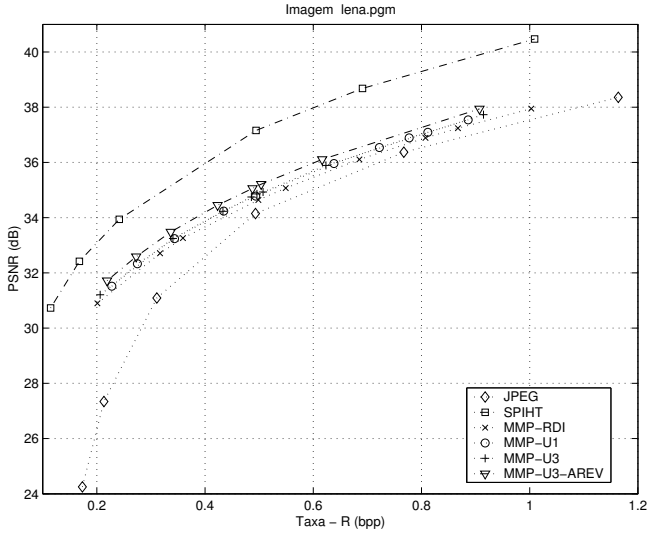
A seguir, temos os resultados com a modificação da atualização aplicada ao método de união com a correção das das estimativas de custos, ou seja, mudamos a forma de atualização do MMP-U3. Sendo assim o método MMP-U3 passou a chamar-se MMP-U3-AREV (MMP-U3 com atualização reversa). Os resultados são apresentados com as curvas taxa-distorção e com uma tabela comparativa para uma taxa fixa igual a 0,5 bpp.

De uma maneira geral as imagens suaves apresentaram resultados bastante expressivos em relação ao método anterior sem atualização reversa, ou seja, o método MMP-U3. Os maiores ganhos foram de 0,3dB (*lena*) e 0,38dB (*barbara*). Para as imagens mistas (*pp1209* e *pp1205*) os resultados foram mantidos.

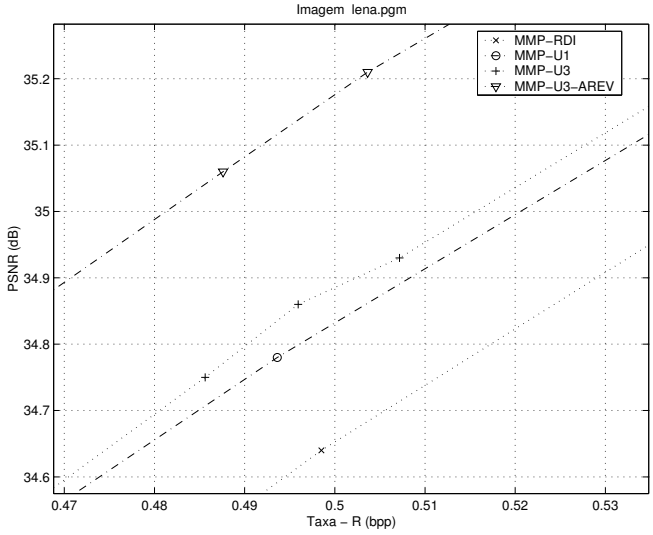
É importante revisar os rótulos usados para as curvas taxa-distorção. Sendo assim, os rótulos são iguais a:

- MMP-U1: Rótulo usado para os resultados do primeiro método de união de blocos chamado de MMP-U1.
- MMP-U3: Rótulo usado para os resultados do método de união de blocos com a análise das estimativas de custos.
- MMP-U3-AREV: Rótulo usado para os resultados do método MMP-U3 com as atualizações reversas.

Na figura 6.11(a) temos as curvas taxa-distorção para a imagem *lena*. Podemos perceber pelas curvas em detalhe da figura 6.11(b) que a imagem *lena* ficou com um ganho de 0,3dB em relação ao método sem atualização reversa, ou seja, o método MMP-U3.



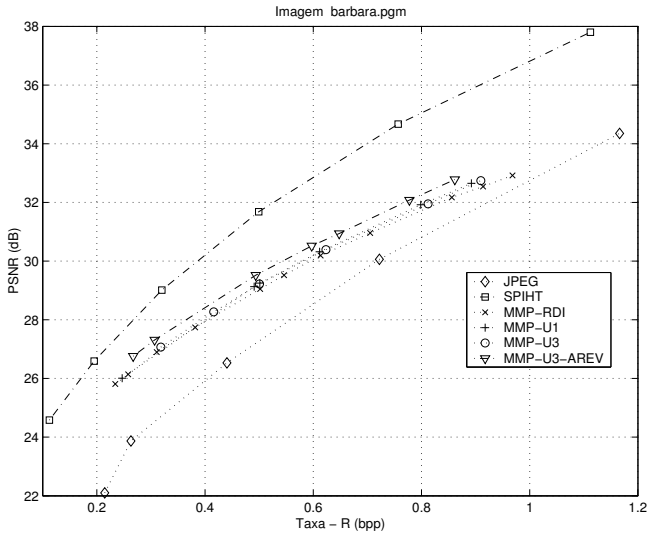
(a) comparativa



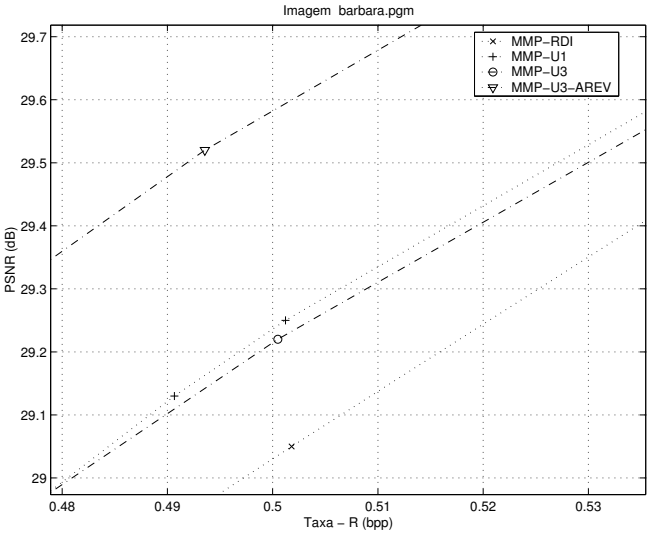
(b) detalhe

Figura 6.11: Curva taxa-distorção para a Imagem *lena*

Na figura 6.12(a) temos as curvas taxa-distorção para a imagem *barbara*. Para esta imagem podemos perceber pelas curvas detalhadas na figura 6.12(b) que seus resultados ficam 0,38dB acima dos resultados apresentados pelo método anterior (MMP-U3).



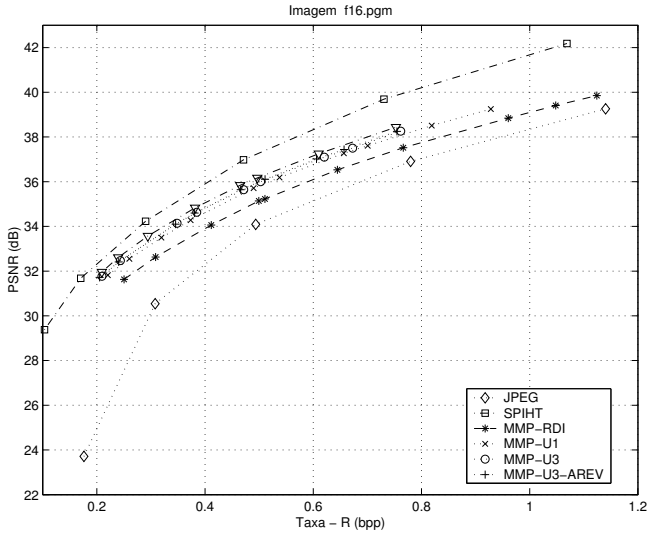
(a) comparativa



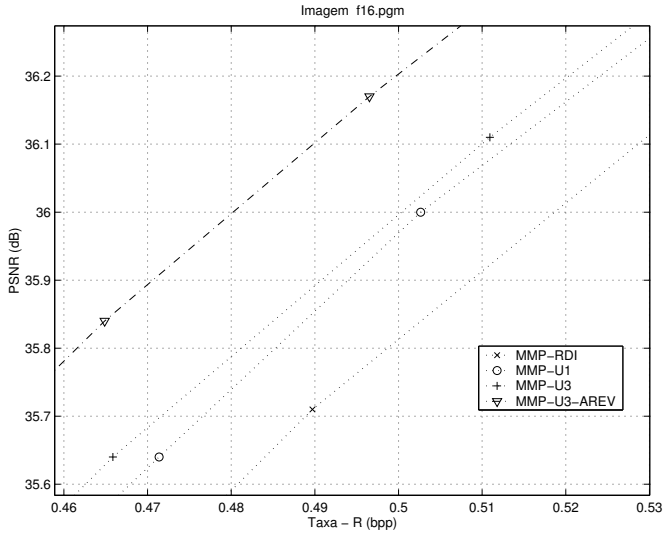
(b) detalhe

Figura 6.12: Curva taxa-distorção para a Imagem *barbara*

A seguir na figura 6.13(a) temos as curvas para a imagem *f16*. Podemos verificar pelo detalhe apresentado na figura 6.13(b) que existe um ganho de 0,25dB em relação ao método MMP-U3.



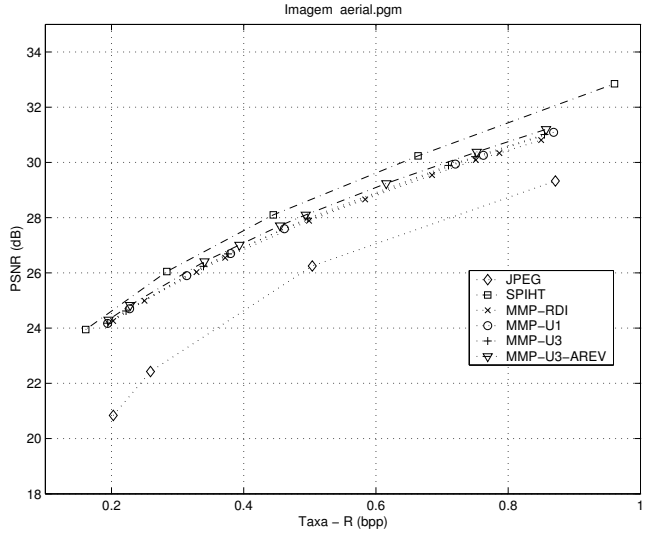
(a) comparativa



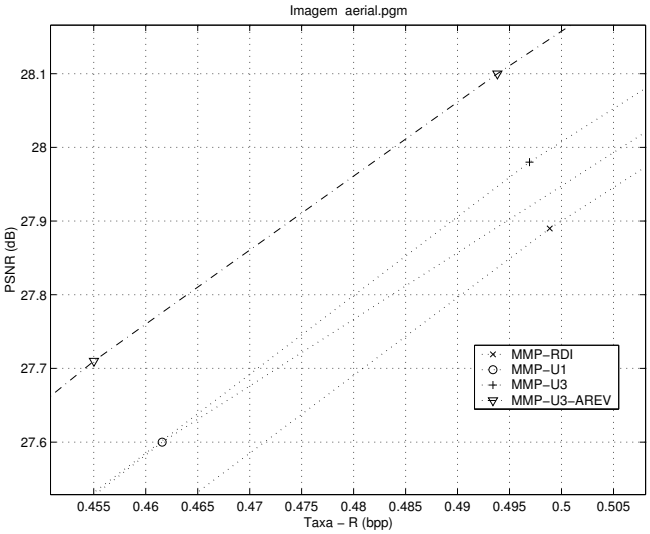
(b) detalhe

Figura 6.13: Curva taxa-distorção para a Imagem *f16*

Na figura 6.14(a) temos as curvas taxa-distorção para a imagem *aerial*. Para esta imagem podemos perceber pelas curvas em detalhe mostradas na figura 6.14(b) que seus resultados ficam 0,2dB acima dos resultados apresentados pelo método MMP-U3.



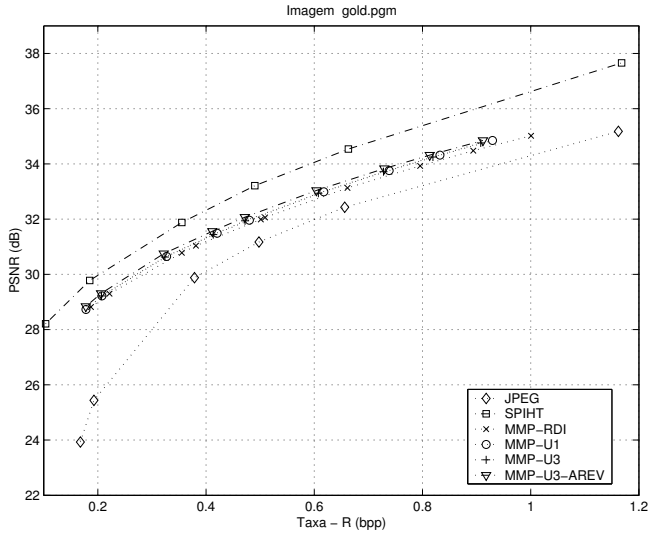
(a) comparativa



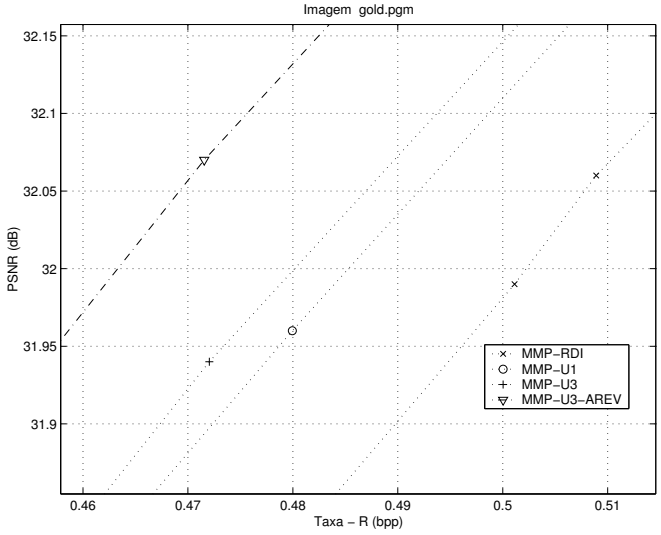
(b) detalhe

Figura 6.14: Curva taxa-distorção para a Imagem *aerial*

Na figura 6.15(a) temos as curvas taxa-distorção para a imagem *gold*. Nesta podemos perceber pelos detalhes mostrados na figura 6.15(b) que seus resultados com ganhos de 0,15dB acima dos resultados apresentados pelo MMP-U3.



(a) comparativa



(b) detalhe

Figura 6.15: Curva taxa-distorção para a Imagem *gold*

6.4 Comentários

A mudança de atualização do dicionário produz resultados satisfatórios. Os maiores ganhos foram atingidos pelas imagens *lena* e *barbara* que ficaram com valores de PSNR em 0,3dB e 0,38dB acima dos valores do método MMP-U3. As imagens *f16* e *aerial* ficaram com ganhos de 0,2dB e a imagem *gold* atingiu 0,15dB (todos em relação aos resultados do método MMP-U3). As imagens *pp1205* e *pp1209* mantiveram seus resultados, em relação ao MMP-U3.

Sendo assim, os resultados obtidos para imagens suaves tiveram um bom acréscimo de PSNR quando comparado ao método de união com a análise das estimativas de custos e as imagens mistas mantiveram seus resultados.

Assim, o método proposto resolve o problema apresentado no capítulo 5, ou seja, o aumento da população de padrões nos dicionários de união melhora o desempenho do método de união.

Capítulo 7

Conclusão

Nesta dissertação, desenvolvemos um método que permite a união de blocos, escolhidos para codificação do esquema de compressão de sinais MMP. Tal desenvolvimento foi alcançado através de três etapas.

A primeira etapa foi realizada no capítulo 4 onde desenvolvemos o método de união de blocos baseado na avaliação de junções à direita e abaixo do bloco analisado, seus resultados tiveram um acréscimo de 0,2dB no geral para imagens suaves em relação ao esquema MMP e para imagens mistas os resultados não foram satisfatórios. Neste método percebemos o problema das *estimativas de custos irreais* isto significa que os custos estimados para o(s) bloco(s) à direita e/ou abaixo não são reais pelo fato da árvore de segmentação não ter processado tais blocos.

No capítulo 5 realizamos a etapa de estudo das *estimativas de custos* onde concluímos que para corrigir tal estimativa era necessária a inversão do sentido de avaliação da união, ou seja, a avaliação de união passa a ser realizada com o bloco da cima e/ou o da esquerda. Os resultados obtidos ficaram em no máximo com 0,1dB de ganho para imagens suaves, contudo para imagens mistas os resultados ficaram próximos dos resultados do MMP.

No capítulo 6 fechamos o método de união com a inclusão de uma atualização do dicionário que é eficiente em um cenário de união, com esta inclusão os resultados tiveram um acréscimo de 0,3dB para as imagens *lena* e *barbara* e para as outras imagens suaves os ganhos foram de 0,2dB.

O desempenho global pode ser visualizado na tabela abaixo. Nesta, temos como referencial, a $PSNR$ para a taxa de 0.5 bpp. As imagens mistam mantiveram seus resultados e os ganhos concentraram-se nas imagens suaves. Os maiores ganhos foram obtidos para as imagens *lena* e *barbara* ficando, aproximadamene, em 0,6dB. A imagem *f16* teve um ganho de 0.4 e as imagens *gold* e *aerial* ficaram com aproximadamente 0,3dB de ganho.

Tabela 7.1: Comparação dos resultados (em dB)

Imagem	MMP-RDI	MMP-U3-AREV	Ganho total
lena	34,60	35,20	0,60
barbara	29,00	29,58	0,58
f16	35,80	36,20	0,40
gold	31,95	32,25	0,30
aerial	27,90	28,16	0,26

Portanto, pelos resultados mostrados, vale a pena usar o método de união de blocos para o esquema de compressão de sinais MMP. Sendo assim, existe uma boa indicação para continuarmos a investigação para este método.

Para trabalhos futuros sugerimos um método de união que permita junções recursivas, ou seja, junções de junções. Um exemplo pode ser visto na figura abaixo.

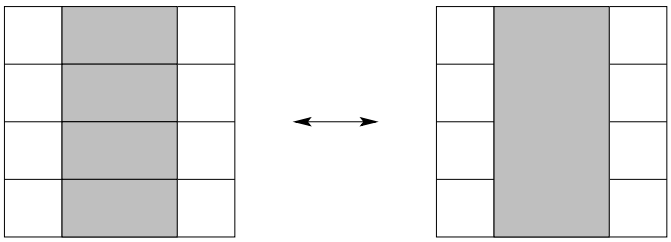


Figura 7.1: Junções de junções

Foi determinado recentemente que métodos que especificam para o MMP um dicionário de estado baseados em críterios de continuidade inter-blocos (SM-MMP) [34] podem levar a uma melhora significativa de relação sinal ruído. Desta forma uma boa direção para desenvolvimentos futuros seria a combinação da classe

de métodos proposta nesta dissertação com críterios de continuidade inter-blocos.

Referências Bibliográficas

[1] SHANNON, C. E., "A Mathematical Theory of Communication", Bell Syst. Tech. Journal, v. 27, 1948.

[2] HUANG, J. J. Y., SCHULTHEISS, P. M., "Block Quantization of Correlated Gaussian Rondom Variables". In: IEEE Trans. Comm., pp. 289–296, 1963.

[3] VIVEK, K. G., "Theoretical Foundations of Transform Coding", IEEE SP Mag., v. 18, n. 5, pp. 9–21, 2001.

[4] VETTERLI, M., KOVACEVIC, J., "Transform Coding: Past, Present and Future", Special Issue IEEE SP Mag., v. 18, n. 5, pp. 9–21, 2001.

[5] PENNENBAKER, W. B., MITCHELL, J. L., JPEG Still Image Data Compression Standard. 1 ed. Van Nostrand Reinhold, 1993.

[6] CLARKE, R. J., Digital Compression of Still Images and Video. Academic Press, 1995.

[7] TAUBMAN, D., MARCELLIN, M., JPEG2000: Image Compressin Fundamentals, Standards and Practice. Kluwer Academic Publishers, 2001.

[8] SHAPIRO, J., "Embedded Image Coding Using Zerotrees of Wavelet Coefficients", IEEE Transactions on Signal Processing, v. 41, pp. 3445–3462, Dec. 1993.

[9] SAID, A., PEARLMAN, W., "A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees". In: IEEE Transactions on Circuits and Systems for Video Technology, v. 6, Chicago, Illinois, June 1996.

[10] "http://www.cipr.rpi.edu/research/SPIHT".

[11] TOPIWALA, P. N., Wavelet Image and Video Compression. 1 ed. Norwell, Kluwer Academic Publishers, 1998.

[12] VETTERLI, M., KOVACEVIC, J., Wavelets and Subband Coding. Prentice Hall, 1995.

[13] STRANG, G., NGUYEN, T., Wavelets and Filter Banks. 1 ed. Wellesley-Cambrige Press, 1996.

[14] CARVALHO, M. B., Compression of Multidimensional Signals based on Recurrent Multiscale Patterns. Tese de d.sc., PEE/Coppe/UFRJ, Rio de Janeiro, RJ, Março 2001.

[15] CARVALHO, M. B., SILVA, E. B., FINAMORE, W. A., "Multidimensional Signals Compression Using Recurrent Patterns". In: Elsevier, pp. 1559–1580, 1995.

[16] HUFFMAN, D. A., "A Method for the Construction of Minimum Redundancy Codes". In: Proceedings of the IRE, v. 40, pp. 1098–1101, 1951.

[17] ABRAMSON, N., Information Theory and Coding. New York, McGraw-Hill, 1963.

[18] ZIV, J., LEMPEL, A., "A Universal Algorithm for Data Compression". In: IEEE Trans. Inf. Theory, v. 23, pp. 337–343, 1977.

[19] ZIV, J., LEMPEL, A., "A Compression of Individual Sequences Via Variable-Rate Coding". In: IEEE Trans. Inf. Theory, v. 24, pp. 530–536, 1978.

[20] The Theory of Error Correcting Codes. New York, North-Holland, 1977.

[21] GERSHO, A., "Quantization", IEEE Comm. Mag., v. 15, pp. 16–29, 1977.

[22] MAX, J., "Quantizing for Minimum Distortion", IRE Trans. Inf. Theory, v. 6, pp. 7–12, 1960.

[23] GERSHO, A., CUPERMAN, V., "A Pattern Matching Technique for Speech Coding", IEEE Communication Magazine, v. 21, pp. 15–21, 1983.

[24] GRAY, R. M., "Vector Quantization". In: IEEE Transaction on Acoustics, Speech and Signal Processing, v. 1, pp. 4–29, 1984.

[25] MAKHOUL, J., ROUCOS, S., GISH, H., "Vector Quantization in Speech Coding". In: Proc. IEEE, n. 73, pp. 1551–1588, 1985.

[26] WITTEN, I., NEAL, R., CLEARY, J. G., "Arithmetic Coding for Data Compression", Comm. ACM, v. 30, pp. 520–540, June 1987.

[27] Processamento Digital de Imagens. Addison-Wesley Publishing Company, 2000.

[28] SAYOOD, K., Introduction to Data Compression. 2 ed. San Francisco, Morgan Kaufmann Publishers, 2000.

[29] JAIN, A. K., Fundamentals of Digital Image Processing. Prentice Hall, 1989.

[30] PAPOULIS, A., Probability Rondom Variables, and Stochastic Processes. 2 ed. McGraw-Hill, 1984.

[31] COVER, T. M., Elements of Information Theory. 2 ed., 1991.

[32] PEEBLES, PEYTON, Z., Probability, Random Variables, and Random Signal Principles. 4 ed. New York, McGraw-Hill, 2001.

[33] FRENDENDALL, G. L., BEHREND, W. L., "Picture Quality - Procedures for Evaluating Subjetive Effects of Interference", IRE, v. 48, pp. 1030–1034, 1970.

[34] FILHO, E. B. L., Compressão de Imagens Utilizando Recorrência de Padrões Multiescala com Critério de Continuidade Inter-blocos. Tese de m.sc., PEE/Coppe/UFRJ, Rio de Janeiro, RJ, Março 2004.

[35] DENN, M. M., Optimization by Variational Methods. 1 ed. New York, McGraw-Hill, 1969.

[36] EVERETT, H., "Generalized Lagrange Multiple Method For Solve Problems of Optimum Allocation Resouces". In: Operation Research, v. 11, pp. 399–417, 1963.

[37] WU, S. W., GERSHO, A., "Rate-Constrained Optimal Block-Adaptative Codging for Digital Tape Recording of HDTV". In: IEEE Trans. on Circuits and System for Video Tech., pp. 100–112, 1991.

[38] SULLIVAN, G. J., BAKER, B. L., "Efficient Quadtree Coding of Imagens and Video". In: IEEE Trans. on Image Processing, v. 3, pp. 327–331, 1994.

[39] KIANG, S. Z., BAKER, R. L., SULLIVAN, G. J., et al., "Recursive Optimal Prune With Applications to Tree Structured Vector Quantizers". In: IEEE Trans. on Image Processing, v. 1, pp. 162–169, 1992.

[40] SHUKLA, DRAGOTTI, MINH, et al., "Rate-Distortion Optimized Tree Structured Compression Algorithms for Piecewise Smooth Images". In: IEEE Transaction Image Processing, 2004.

[41] RAMCHANDRAN, K., VETTERLI, M., "Best Wavelet Packet Bases in a Rate Distortion Sense". In: IEEE Transaction Image Processing, pp. 160–175.

Apêndice A

Métodos de Podagem-União

Neste apêndice mostramos os *métodos de podagem* e *podagem-união* desenvolvidos em [40]. Na sessão A.1 temos o *algoritmo de podagem* que fornece uma árvore binária segmentada otimizada no sentido taxa-distorção e na sessão A.2 temos o algoritmo que avalia e codifica as possiveis uniões que podem ser realizadas.

A.1 Algoritmo de Podagem (Prune)

Passo 1 Inicialização

1.1 Segmente o sinal de entrada usando a decomposição em árvore binária até uma profundidade J.

1.2 Aproxime cada nó por um polinômio $p(t)$ de grau $\leq P$ em relação ao menor erro quadrático.

1.3 Gere a curva R-D para cada nó aproximando o nó pelo polinômio quantizado $\widehat{p}(t)$ que é obtido pela quantização escalar dos coeficientes de $p(t)$.

Passo 2 Podagem baseada no custo lagrangeano

2.1 Para um dado λ efetue a *poda* ou *descarte* do nó filho se a soma dos custos lagrangeanos é maior que ou igual ao custo lagreangeano do nó pai. Ou seja: $\{D_{f1} + D_{f2} + \lambda(R_{f1} + R_{f2})\} \leq \{D_p + \lambda R_p\}$. Onde $f1, f2$ são nós folhas e p indica o nó pai.

2.2 Determine D^* e R^* somando-se todas as taxas/distorções de todas folhas da sub-árvore podada.

Passo 3 Busca da inclinação λ dasejada

3.1 O valor de λ é determidado iterativamente até a taxa constante R_o. Esse algoritmo pode ser encontrado em [41] sendo dado pelos passos abaixo:

3.2 Determine λ_{min} e λ_{max} tal que $R^*(\lambda_{max}) \leq R_o \leq R^*(\lambda_{min})$. **Se** obtivermos uma igualdade **então Retorne** e o valor de λ estara determinado. **Se não** siga para o passo *3.3*.

3.3 Calcule $\lambda_x = \frac{D^*(\lambda_{min}) - D^*(\lambda_{max})}{R^*(\lambda_{max}) - R^*(\lambda_{min})}$

3.4 Execute o **Passo 2** para λ_x.

Se $(R_o = R^*(\lambda_x))$ **então** pare.

Se não Se$(R_o < R^*(\lambda_x))$ faça $\lambda_{min} = \lambda_x$ e execute *3.3*.

Se não faça $\lambda_{max} = \lambda_x$ e execute *3.3*.

A.2 Algoritmo de Podagem-União (Prune-Join)

Passo 1 Inicialização

1.1 Execute o **Passo 1** e o **Passo 2** do *algoritmo de podagem* para encontrar a melhor árvore podada para um dado λ.

Passo 2 Junção de vizinhos

2.1 Dada uma árvore podada, determine se uma junção pode ser codificada de acordo com o custo lagrangeano, ou seja, $\{D_{f1} + D_{f2} + \lambda(R_{f1} + R_{f2})\} \leq \{D_j + \lambda R_j\}$. Onde $f1, f2$ indicam os vizinhos testados e j indica o bloco junção.

Passo 3 Busca da inclinação λ dasejada

3.1 Execute o **Passo 3** do *algoritmo de podagem.*

Apêndice B

Imagens

Neste apêndice encontramos as imagens *lena, barbara, aerial, pp1205, pp1209, f16* e *gold* usadas para as simulações.

B.1 Imagem *Lena*

Figura B.1: Imagem *lena*

B.2 Imagem *Barbara*

Figura B.2: Imagem *barbara*

B.3 Imagem *aerial*

Figura B.3: Imagem *aerial*

B.4 Imagem *pp1205*

Figura B.4: Imagem *pp1205*

B.5 Imagem *pp1209*

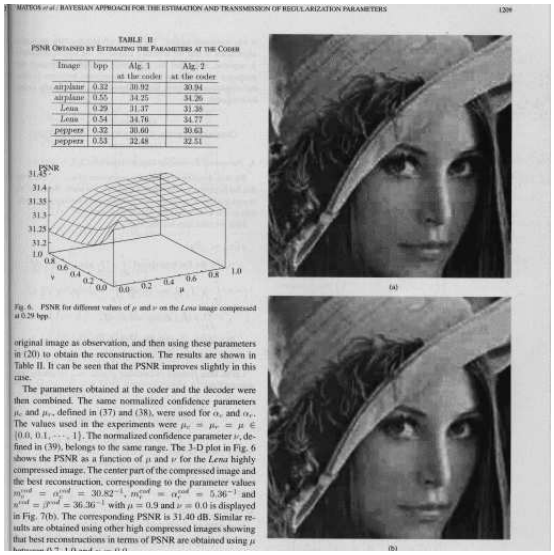


Figura B.5: Imagem *pp1209*

B.6 Imagem *f16*

Figura B.6: Imagem *f16*

B.7 Imagem *gold*

Figura B.7: Imagem *gold*