COPPE/UFRJ

CODIFICAÇÃO DE VÍDEO VIA CASAMENTO DE PADRÕES USANDO ESTIMAÇÃO DE MOVIMENTO

Diego Felix de Souza

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

Orientador: Eduardo Antônio Barros da Silva

Rio de Janeiro
Maio de 2010

CODIFICAÇÃO DE VÍDEO VIA CASAMENTO DE PADRÕES USANDO ESTIMAÇÃO DE MOVIMENTO

Diego Felix de Souza

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

Examinada por:

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

Prof. Gelson Vieira Mendonça, Ph.D.

Prof. Lisandro Lovisolo, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MAIO DE 2010

Souza, Diego Felix de

Codificação de vídeo via casamento de padrões usando estimação de movimento/Diego Felix de Souza. – Rio de Janeiro: UFRJ/COPPE, 2010.

XVII, 88 p.: il.; 29, 7cm.

Orientador: Eduardo Antônio Barros da Silva

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2010.

Referências Bibliográficas: p. 45 – 46.

1. Codificação de vídeo. 2. Casamento de padrões. 3. Padrão H.264/AVC. I. Silva, Eduardo Antônio Barros da. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Ao meu irmão.

Agradecimentos

Gostaria de agradecer minha filha por existir. À minha mãe por ter esquecido o anticoncepcional, ao meu pai por ser contra o aborto e toda a minha família que mesmo sem ter sido desejado me amou como se tivesse sido.

A todos os amigos espalhados pelo Brasil por manterem minha insanidade controlada. Principalmente aos amigos que fiz nesses últimos dois anos entre eles Brenno Strutt, César Bouças, Ícaro Souza, Pietro Grassia, Luiz Menegaz, Gabriel Matos, Camila Gussen, Markus Lima, Tadeu Ferreira e toda a equipe do LPS. Ao José Fernando por tirar minhas dúvidas em C e Linux. Acrescento ainda um agradecimento especial para Alessandro Dutra por apoio, carisma e palavras de incentivo.

À companhia aérea Azul, aos criadores e organizadores da Oktoberfest e ao estilo de vida da sociedade manauara.

Por último, mas não menos importante, agradeço ao meu orientador, o Professor Eduardo Antônio Barros da Silva e ao povo brasileiro que pagando impostos financiou este projeto através do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

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

CODIFICAÇÃO DE VÍDEO VIA CASAMENTO DE PADRÕES USANDO ESTIMAÇÃO DE MOVIMENTO

Diego Felix de Souza

Maio/2010

Orientador: Eduardo Antônio Barros da Silva

Programa: Engenharia Elétrica

A estimação de movimento no padrão H.264/AVC é responsável por uma fatia significativa de seu desempenho. Ela mistura diversos tamanhos de bloco, vários quadros de referência e interpolação de até 1/4 de pixel para a componente de luma. Assim sendo, pode-se considerar a estimação de movimento no H.264/AVC um método poderoso de casamento de padrões. Nesta dissertação, se pretende avaliar o quanto é viável usar apenas a estimação de movimento sem codificar o resíduo da predição.

Na primeira parte desta dissertação, será mostrado como é possível melhorar o codificador H.264/AVC eliminando do mesmo as transformadas e, consequentemente, a quantização. Isto é feito descartando seletivamente o resíduo da predição inter, desta forma, mantendo a compatibilidade com a norma H.264/AVC.

Na segunda parte, abrindo mão da compatibilidade com a norma H.264/AVC, é criado um novo codificador de vídeo com base apenas no casamento de padrões propiciado pela estimação de movimento. Isto é feito inserindo blocos de até 1×1 pixel na estimação de movimento e descartando os resíduos de predição.

Os resultados experimentais mostram que em alguns casos é viável o descarte dos resíduos, e que vale a pena aprofundar a investigação do paradigma proposto.

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

PATTERN MATCHING VIDEO CODING USING MOTION ESTIMATION

Diego Felix de Souza

May/2010

Advisor: Eduardo Antônio Barros da Silva

Department: Electrical Engineering

The motion estimation in the H.264/AVC standard is responsible for a significant portion of its performance. It mixes various block sizes, multiple reference pictures and interpolation with up to quarter-sample resolution for the luma component. Thus, the motion estimation on H.264/AVC can be considered a powerful pattern matching method. In this dissertation, it is assessed how viable is to use only the motion estimation without coding the residual data.

In the first part of this dissertation, it will be shown how the H.264/AVC encoder can be improved by eliminating the transforms and, therefore, the quantisation. This is done by selectively discarding the residue of inter prediction, thus keeping the compatibility with H.264/AVC standard.

In the second part, the need for compatibility with the H.264/AVC standard is waived and a new video encoder is proposed based only on the pattern matching provided by the motion estimation. This is done by inserting blocks of up to 1×1 pixel in the motion estimation and discarding all the residual data.

The experimental results show that in some cases it is advantageous to discard the residual data, and that it is worth a further investigation of the proposed paradigm.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AVC	*Advanced Video Coding*, p. 3
CABAC	*Context-based Adaptive Binary Arithmetic Coding*, p. 12
CAVLC	*Context-based Adaptive Variable Length Coding*, p. 12
CBP	*Coded Block Pattern*, p. 13
CPST	Casamento de Padrões Sem Transformadas, p. 27
DCT	*Discrete Cosine Transform*, p. 11
DRSB	Descarte de Resíduos em *Slices* B, p. 13
FRExt	*Fidelity Range Extensions*, p. 3
GOP	*Group Of Pictures*, p. 22
ISDB-TB	*Integrated Services Digital Broadcasting Terrestrial Brazil*, p. 3
ISO/IEC	*International Organization for Standardization/International Electrotechnical Commission*, p. 3
ITU-T	(*International Telecommunications Union - Telecomunicati-ons*, p. 3
I_PCM	*Intra Pulse Code Modulation*, p. 7
JVT	*Joint Video Team*, p. 3
MMP	*Multi-dimensional Multiscale Parser*, p. 4
MPEG	*Moving Picture Experts Group*, p. 3
MSE	*Mean Squared Error*, p. 15
OCSB	Otimização na Codificação de *Slices* B, p. 21

PSNR *Peak Signal to Noise Ratio*, p. 15

QP *Quantization Parameter*, p. 14

SAD *Sum of Absolute Differences*, p. 8

SATD *Sum of Absolute Transformed Differences*, p. 9

SD *Standard Definition*, p. 1

SE *Syntax Element*, p. 12

SSE *Sum Square of Errors*, p. 9

VCEG *Video Coding Experts Group*, p. 3

Capítulo 1

Introdução

Nesse capítulo, será mostrado um resumo sobre vídeo digital e sobre a história do padrão H.264/AVC. O padrão de codificação de vídeo H.264/AVC, como será indicado adiante, é o padrão que foi utilizado como base no desenvolvimento dessa dissertação. Logo após, será feito um resumo sobre casamento de padrões e, em seguida, a explicação sobre o objetivo e a organização desta dissertação.

1.1 Vídeo Digital

Uma sequência de vídeo é um sinal tridimensional com uma dimensão no tempo e as outras duas no espaço. Quando o espaço e tempo são discretizados, obtêm-se uma sequência de vídeo digital, onde cada ponto desse sinal tridimensional representa um pixel. Para que um computador consiga reproduzir o valor de um pixel, a informação contida nele precisa ser binarizada. Assim, geralmente um pixel contém um valor que representa digitalmente a componente de luminância, chamada de luma (Y´) e dois valores para representar digitalmente a componente de crominância, chamada de croma (Cb ou U e Cr ou V).

Levando em conta a percepção humana, 256 valores diferentes para cada componente de luma e de croma são suficientes para representar um pixel, o que corresponde a 24 bits por pixel. Como os humanos são mais perceptíveis a variações de luminância do que crominância, é possível subamostrar as componentes de croma, sem perdas visíveis na qualidade. O padrão mais comum de subamostragem das componentes de croma é o padrão 4:2:0, onde apenas existem dois valores para a componente de croma (U e V) para quatro valores para a componente de luma (Y´), como mostra a Figura 1.1.

No padrão de subamostragem 4:2:0 são necessários 12 bits por pixel para representar uma imagem. Logo, em um quadro digital com essa subamostragem de cor, com resolução SD (*Standard Definition*) de 720×480 pixeis, são necessários 4147200 bits. Portanto, numa transmissão usual de 30 quadros por segundo para

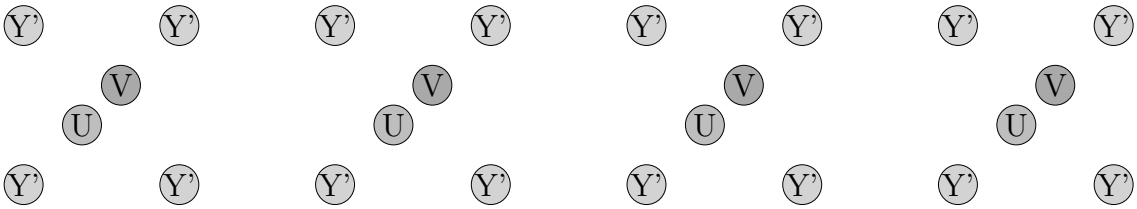


Figura 1.1: Padrão de subamostragem 4:2:0

uma sequência de vídeo com essa resolução é necessária uma taxa de 124,416 Mbps, que é extremamente alta. Assim, é imprescindível uma compressão de qualquer sinal de vídeo digital.

Compressão de vídeo ou codificação de vídeo é o processo que realiza uma compressão com perdas de uma sequência de vídeo digital e a transforma em uma sequência de bits (*bitstream*) para armazenamento ou transmissão. Essa sequência de bits comprimida pode ser decodificada em uma sequência de vídeo digital novamente, porém como geralmente a compressão é feita com perdas, a sequência de vídeo digital obtida possui uma diferença nos valores de pixeis em relação à original. Quanto maior a compressão, maiores serão as perdas e portanto, maior será a distorção ou o erro obtido quadro a quadro em relação ao original. Definindo a taxa como a quantidade de bits para a transmissão ou armazenamento de uma sequência de vídeo digital, temos que um codificador pode ser considerado melhor que outro quando o mesmo obtém uma menor taxa com a mesma distorção ou uma menor distorção com a mesma taxa.

A curva taxa-distorção mostra a distorção obtida para cada valor de taxa de um determinado codificador. Assim, como esperado, quanto maior a taxa menor será a distorção e quanto menor a taxa maior será a distorção, como pode ser visualizado na Figura 1.2a. A curva taxa-distorção pode ser utilizada para determinar qual codificador é mais eficiente apenas através da observação das curvas taxa-distorção de cada codificador. Por exemplo, na Figura 1.2b, o codificador que produziu a curva mais próxima dos eixos (linha contínua) é mais eficiente do que o codificador que gerou a curva mais afastada dos eixos (linha pontilhada) em todas as taxas.

1.2 História do Padrão H.264/AVC

Desde o lançamento do padrão H.261 em 1991, diversos codificadores de vídeo foram desenvolvidos ao longo dos anos, acompanhando o desenvolvimento do *hardware* de cada época. Entre os diversos padrões de compressão de vídeo que já foram desenvolvidos até hoje, os mais utilizados foram os padrões H.261, MPEG-1, H.262/MPEG-2 (utilizado nos DVDs e em diversos sistemas de TV digital), H.263 e MPEG-4 Part 2. O estado da arte em codificação de vídeo atualmente foi desenvolvido pelo JVT

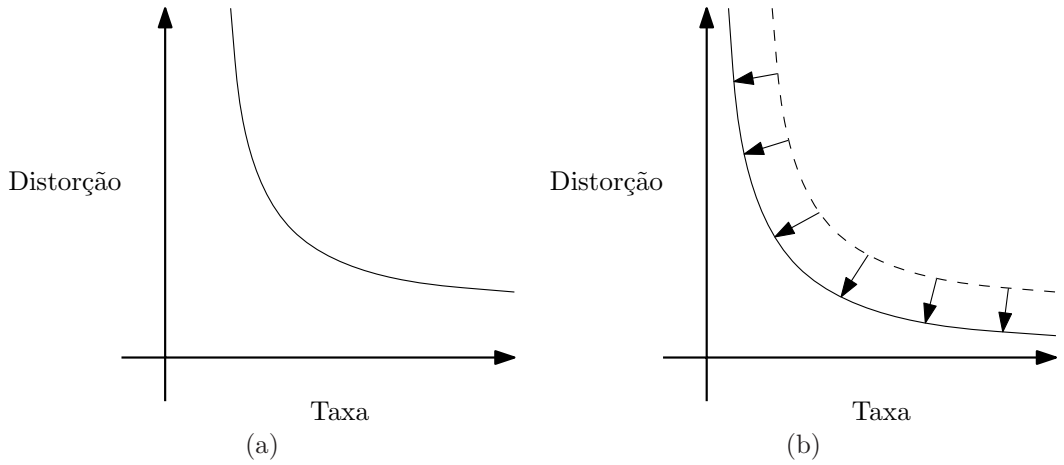


Figura 1.2: Curvas taxa-distorção: 1.2a exemplo de uma curva taxa-distorção e 1.2b curva taxa-distorção almejada de um codificador mais eficiente

(*Joint Video Team*), um grupo formado por dois outros grupos o VCEG (*Video Coding Experts Group*) e o MPEG (*Moving Picture Experts Group*). O grupo VCEG é formado por especialistas da ITU-T (*International Telecommunications Union - Telecomunications*) e o grupo MPEG é formado por especialistas da ISO/IEC (*International Organization for Standardization/International Electrotechnical Commission*). A ITU e a ISO/IEC são duas organizações internacionais responsáveis por normatizações em diversas áreas. Assim, em 2003, o JVT publicou a norma ITU-T H.264/MPEG-4 (Part 10) *Advanced Video Coding*, o atual estado da arte em codificação de vídeo, alcançando em algumas sequências o dobro de compressão em relação ao H.262/MPEG-2.

A norma do padrão ITU-T H.264/MPEG-4 (Part 10) *Advanced Video Coding* (mais conhecida como H.264/AVC) define diversas ferramentas de codificação de vídeo que são divididas em 3 perfis: *Baseline*, *Main* e *Extended*. A divisão dessas ferramentas em perfis facilita a interoperabilidade em diversas aplicações. Cada perfil ainda é dividido em 15 níveis que definem os limites de tamanho de quadro, taxa de decodificação, tamanho da memória de armazenamento de quadros entre outros. Em Julho de 2004, a emenda FRExt (*Fidelity Range Extensions*) [1] [2] adiciona ao padrão H.264/AVC um conjunto de melhoramentos que aumentam ainda mais o desempenho do codificador H.264/AVC. Entre esses melhoramentos estão o uso de transformadas 8×8 e subamostragem de cor nos padrões 4:4:4 e 4:2:2, já que antes só era possível a subamostragem de cor 4:2:0. Assim, a emenda FRExt adiciona 4 novos perfis com base no perfil *Main*, chamados de perfis do tipo *High*: *High*, *High 10*, *High 4:2:2* e *High 4:4:4* [2]. A emenda também adiciona um nível, totalizando 16 níveis para cada perfil na norma H.264/AVC. Durante esta dissertação será utilizado o perfil *High* e o nível 4.0, a mesma configuração utilizada pelo padrão ISDB-TB (*Integrated Services Digital Broadcasting Terrestrial Brazil*), padrão que

normatiza a transmissão do sinal de TV digital terrestre no Brasil.

Uma poderosa ferramenta responsável pelo excelente desempenho do codificador H.264/AVC é a estimação/compensação de movimento usando blocos de tamanhos variáveis e múltiplas referências, que será melhor explicada no Capítulo 2. O processo de estimação de movimento, basicamente, tenta prever um bloco de pixeis realizando uma busca nos quadros já processados. Uma das informações de movimento que será codificada pelo processo de compensação de movimento é a diferença entre o bloco original e o predito através de transformadas e quantizações. Assim, se eliminarmos essa diferença do codificador a estimação/compensação de movimento funcionará como um método de casamento de padrões. E quanto mais poderosa for a estimação/compensação de movimento, melhor será o método de casamento de padrões. A questão que esta dissertação investiga é: a estimação/compensação de movimento pode ser tão boa a ponto de dispensar a codificação do resíduo da predição?

1.3 Casamento de Padrões

Por casamento de padrões, entende-se como uma técnica de codificação de um sinal de qualquer dimensão através de uma palavra ou índice de um dicionário com a mesma dimensão do sinal. Assim, ao receber um sinal, o processo de casamento de padrões efetua uma busca no dicionário por um sinal que mais se aproxime do sinal original de acordo com alguma métrica. Após encontrar o elemento do dicionário mais próximo ao sinal, o processo de casamento de padrões irá transmitir ou armazenar apenas a palavra ou índice desse elemento. Um exemplo claro de utilização desse processo é a quantização vetorial [3].

Essa é uma visão geral de casamento de padrões, há diversas formas do mesmo ser implementado, principalmente na criação do dicionário e de suas palavras. Por exemplo, o dicionário poderá ser adaptativo ou fixo, com o dicionário adaptativo a cada sinal processado o dicionário é atualizado, crescendo até um limite. Mas a forma de atualização e o tamanho máximo desse dicionário podem ser criados de diversas maneiras. Um exemplo disso é o algoritmo de casamento de padrões MMP (*Multi-dimensional Multiscale Parser*) [4].

Assim, se eliminarmos as transformadas e quantizações do codificador H.264/AVC obteremos um codificador de vídeo utilizando apenas casamento de padrões através da estimação de movimento e da compensação de movimento. O dicionário desse casamento de padrões são os quadros já processados anteriormente e as palavras do dicionário serão as informações de movimento que identificam cada bloco de pixel nos quadros armazenados. A atualização desse dicionário será feita de forma automática à medida que os quadros forem processados e o tamanho do

dicionário estará diretamente relacionado com a faixa de busca e o número de quadros que o codificador H.264/AVC é capaz de armazenar. A faixa de busca é uma área dos quadros já processados que serão utilizadas como base para o processo da estimação de movimento, o qual será explicado com mais detalhes no Capítulo 2.

1.4 Objetivo e Organização da Dissertação

O objetivo desta dissertação é verificar o quão viável é a codificação de vídeo por casamento de padrões, utilizando apenas a estimação e compensação de movimento para realizar essa operação. Dessa forma, serão descartadas as transformadas e quantizações. Primeiramente, será mostrada e implementada uma forma de codificação por casamento de padrões ainda compatível com a norma H.264/AVC, utilizando apenas a estimação e compensação de movimento. Depois será criado um novo codificador de vídeo com base completamente em casamento de padrões utilizando uma estimação de movimento "refinada", onde serão acrescentados blocos com tamanhos de até 1 pixel.

No Capítulo 2, será feita uma breve revisão sobre o padrão H.264/AVC perfil *High*. Informações detalhadas podem ser encontradas na própria norma [1].

No Capítulo 3, será mostrado como é possível melhorar o codificador H.264/AVC excluindo do mesmo as transformadas e, consequentemente, a quantização. Isso será realizado de duas formas: uma eliminando completamente o resíduo da predição inter e outra eliminando seletivamente este resíduo. Nas duas formas, ao suprimir o resíduo, o codificador H.264/AVC estará funcionando como um codificador de vídeo por casamento de padrões, entretanto, as modificações sugeridas nesse capítulo ainda são compatíveis com a norma H.264/AVC.

No Capítulo 4, será mostrada a criação de um codificador de vídeo que utiliza somente a operação de casamento de padrões, eliminando completamente as transformadas e quantizações. Como existem diversas maneiras para implementar esse casamento de padrões, a escolhida foi utilizar uma modificação na estimação de movimento e compensação de movimento do H.264/AVC para realizar a operação de casamento de padrões. A modificação da estimação e compensação de movimento é a inserção de blocos menores que 4×4 pixeis e a completa eliminação do resíduo da predição.

Por fim serão apresentadas as conclusões a respeito desta dissertação no Capítulo 5, assim como algumas sugestões para trabalhos futuros.

Capítulo 2

Codificador de Vídeo H.264/AVC

Neste capítulo, serão esboçados alguns conceitos relativos ao decodificador e codificador de vídeo H.264/AVC, necessários para o entendimento da ideia proposta nesta dissertação. Informações detalhadas a respeito do padrão H.264/AVC podem ser obtidas através da norma H.264/AVC [1]. Na Seção 2.1, será apresentada uma ideia geral do codificador H.264/AVC. Em seguida, será apresentada com mais detalhes a estimação de movimento empregada no codificador, na Seção 2.2. Por fim, será apresentado o codificador aritmético usado pelo codificador H.264/AVC na Seção 2.3.

A codificação de vídeo no padrão H.264/AVC [1] consiste em dividir um quadro em vários blocos menores, de 16×16 pixeis, denominados macroblocos e codificá-los individualmente. Assim, cada quadro é dividido em um ou mais *slices*, onde um *slice* é um conjunto de macroblocos, que pode conter de 1 até o total de macroblocos de um quadro (um *slice* por quadro). Cada *slice* é codificado com técnicas específicas e com uma certa independência entre si para evitar propagação de erros. Dentre as várias opções à disposição do codificador [5] [2], a predição do macrobloco a ser codificado a partir de um ou dois quadros de referência é o processo de maior interesse para este trabalho e será objeto de descrição mais detalhada a seguir.

A Figura 2.1 mostra um diagrama de blocos simplificado da codificação de um macrobloco. O bloco Decisão do Modo é responsável por escolher a melhor predição para o macrobloco a ser processado. Em seguida, a diferença entre o macrobloco predito e o original – o resíduo – é transformada e quantizada nos blocos T e Q respectivamente. Por fim, o bloco Codificação por Entropia irá codificar os resíduos e as informações relativas a predição.

2.1 Conceitos Básicos

Há duas formas de predição no padrão H.264/AVC, denominadas Inter e Intra. No modo Intra, apenas os dados pertencentes ao *slice* atual são considerados para a

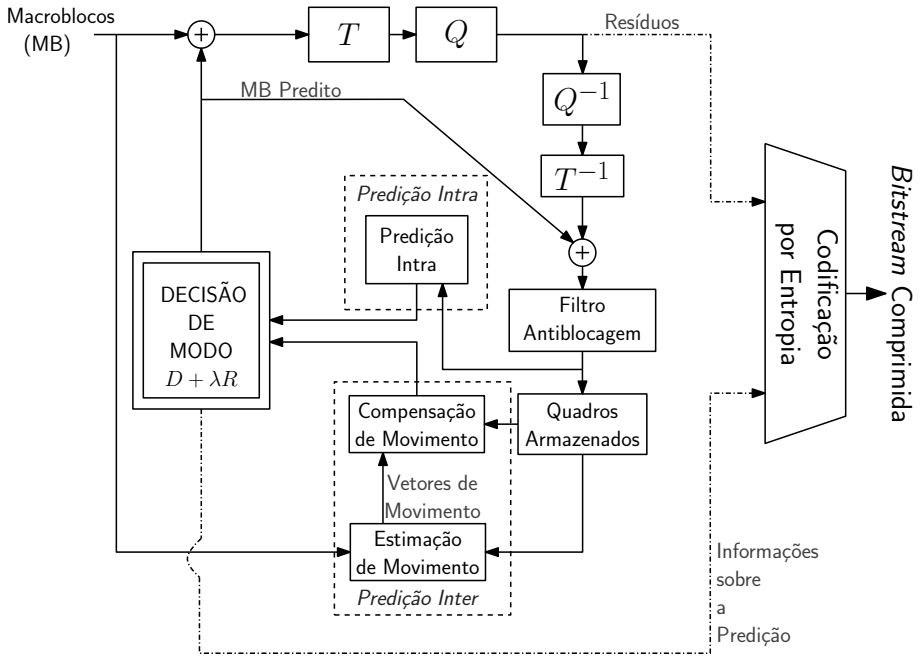


Figura 2.1: Diagrama de blocos de um codificador H.264/AVC simplificado

predição de um determinado (macro)bloco. Para os dados de luma, há 4 modos distintos de predição disponíveis para blocos de 16×16 pixeis e 9 modos para blocos de 4×4 e 8×8 pixeis. No caso dos dados de croma, há 4 modos possíveis de predição, idênticos aos definidos para os blocos 16×16 pixeis de luma.

Outra forma de codificação de macroblocos, considerada como Intra, é a I_PCM (*Intra Pulse Code Modulation*), onde o valor de cada pixel de todas as componentes – luma e croma – são transmitidos diretamente. Nota-se que esse é o pior tipo de codificação, mesmo tendo distorção nula. É por esse motivo que a quantidade de macroblocos I_PCM pode ser utilizada para indicar a ineficiência de um codificador.

Durante a predição do tipo Inter, apenas um ou dois quadros anteriormente decodificados podem ser usados como referência para os blocos a serem processados, desde 16×16 até 4×4 pixeis. Na sequência de vídeo, a posição temporal dos quadros de referência é irrelevante, podendo os mesmos estarem tanto no passado como no futuro em relação ao bloco sendo codificado. A Figura 2.2 mostra alguns exemplos de predições Inter.

Os *slices* são divididos em três tipos distintos: *intra* (I), *predictive* (P) e *bi-predictive* (B). Como dito anteriormente, para cada tipo de *slice*, há um conjunto de métodos que podem ser usados para calcular o macrobloco de referência. Por exemplo, o codificador somente utilizará a predição Intra para *slices* do tipo I. Já para os *slices* dos tipos P e B é possível utilizar tanto a predição Intra como a Inter.

A fim de serem utilizados na predição Inter, quadros anteriormente decodificados são armazenados em duas listas, chamadas de lista 0 e lista 1. Na predição Inter

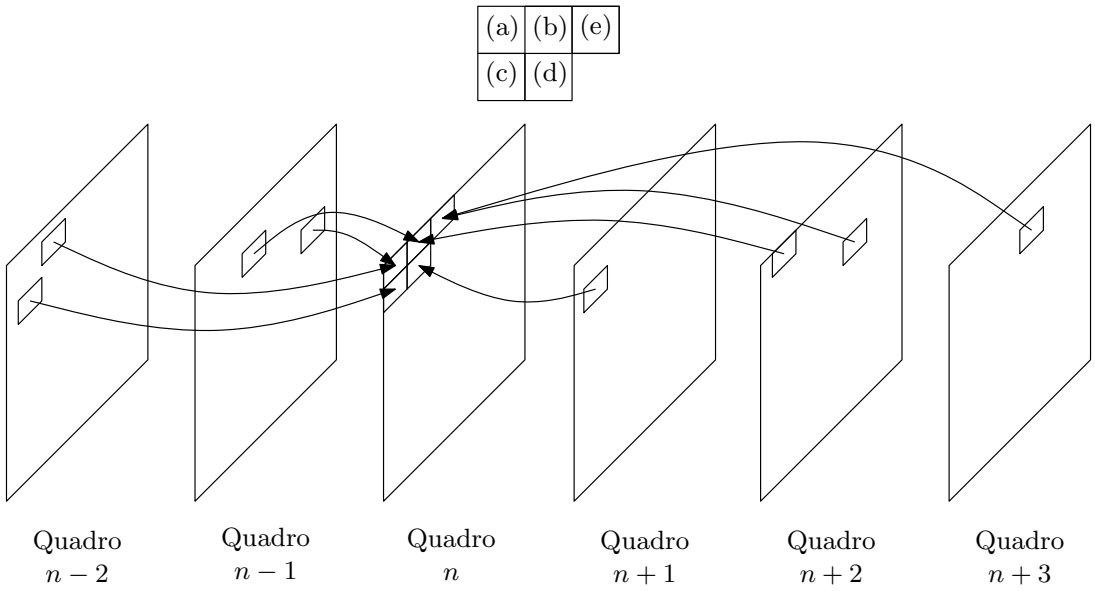


Figura 2.2: Exemplos de predições Inter: (a) duas referências do passado; (b) uma do passado e outra do futuro; (c) uma do passado; (d) uma do futuro; (e) duas do futuro

realizada nos *slices* do tipo P, ou predição unidirecional, somente quadros armazenados na lista 0 são utilizados. Já na predição Inter dos *slices* do tipo B, ou predição bidirecional, quadros armazenados na lista 0 e/ou na lista 1 são empregados. A determinação do macrobloco ou bloco de referência é feita através da estimação de movimento.

2.2 Estimação de Movimento

Na predição unidirecional, como dito anteriormente, a estimação de movimento é realizada com base nos quadros armazenados na lista 0. Assim, o codificador procura, nestes quadros, por um bloco de referência que gere o menor erro de predição. A norma não especifica qual algoritmo de busca ou métrica deverá ser escolhida, portanto, cada codificador é projetado para utilizar o método de busca e métrica que achar mais conveniente. Para o cálculo do erro, as seguintes funções são definidas como exemplo ($o_{x,y}$ e $p_{x,y}$ representam respectivamente os pixeis originais e preditos; $O_{n,m}$ e $P_{n,m}$ representam os coeficientes da transformada de Hadamard dos respectivos blocos originais e preditos) [6]:

- A soma dos erros absolutos (*Sum of Absolute Differences* - SAD)

$$Erro = \sum_{x,y} |o_{x,y} - p_{x,y}| \tag{2.1}$$

- A soma dos erros quadráticos (*Sum of Squared Errors* - SSE)

$$Erro = \sum_{x,y} (o_{x,y} - p_{x,y})^2 \tag{2.2}$$

- soma dos erros transformados absolutos (*Sum of Absolute Transformed Differences* - SATD)

$$Erro = \sum_{n,m} |O_{n,m} - P_{n,m}| \tag{2.3}$$

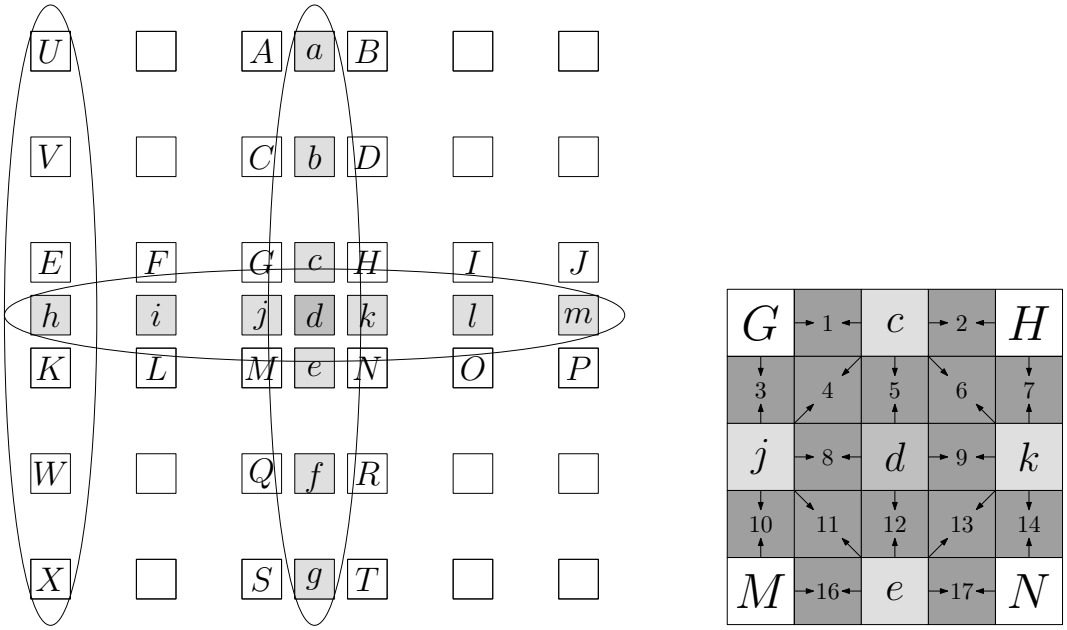
A fim de permitir a estimação de movimento com resolução menor que um pixel, a norma [1] [6] define filtros de interpolação para as componentes de luma e croma, com resoluções máximas de 1/4 e 1/8 (subamostragem de cor 4:2:0) de pixel respectivamente. Para a componente luma, as amostras de meio pixel são geradas primeiro, através de um filtro FIR com coeficientes iguais a [1 -5 20 20 -5 1]/32. Em seguida, são geradas as amostras de 1/4 de pixel através de uma interpolação linear. A Figura 2.3 representa como são geradas as interpolações de 1/2 pixel e 1/4 de pixel, onde A, B, C... representam os pixeis originais; a, b, c... representam as amostras de 1/2 pixel e 1, 2, 3... representam as amostras de 1/4 de pixel. Primeiramente, são calculadas as amostras de 1/2 pixel adjacentes – horizontalmente ou verticalmente – a duas amostras de pixeis, por exemplo, as amostras c, e, j e k na Figura 2.3. Em seguida, são calculadas as amostras de meio pixel restantes – que estão localizadas na direção diagonal entre pixeis – utilizando as amostras de meio pixel previamente calculadas, por exemplo, a amostra d na Figura 2.3. Assim, amostras de meio pixel são geradas interpolando as amostras de pixeis ou meio pixel mais próximas, horizontais ou verticais. Por exemplo, as amostras de meio pixel h e c são geradas interpolando os pixeis verticais e horizontais respectivamente:

$$h = round((U - 5V + 20E + 20K - 5W + X)/32) \tag{2.4}$$

e

$$c = round((E - 5F + 20G + 20H - 5I + J)/32), \tag{2.5}$$

onde a função $round()$ é o arredondamento para o inteiro mais próximo. As amostras de meio pixel que estão na direção diagonal entre pixeis, como a amostra de meio pixel d na Figura 2.3, são geradas interpolando amostras de meio pixel horizontais ou verticais (o resultado será o mesmo). Por exemplo, na Figura 2.3a, a amostra de meio pixel d pode ser gerada interpolando as amostras de pixel a, b, c, e, f e g ou h, i, j, k, l e m. Amostras de um quarto de pixel são geradas através de uma interpolação linear das duas amostras – de pixel ou meio pixel – mais próximas de



(a) Interpolação de meio pixel

(b) Interpolação de um quarto de pixel

Figura 2.3: Interpolações para componente de luma: 2.3a nas posições de 1/2 pixel e 2.3b nas posições de 1/4 de pixel

acordo com a Figura 2.3b.

Para as componentes de croma, com subamostragem 4:2:0, amostras com resolução de 1/8 de pixel são geradas através da interpolação linear dos 4 pixeis mais próximos. Como mostrado na Figura 2.4, onde A, B, C e D são os pixeis de croma mais próximos e a é a amostra de 1/8 de pixel de croma a ser calculada de acordo com a seguinte expressão:

$$a = round([(8-x)(8-y)A + (x)(8-y)B + (8-x)(y)C + (x)(y)D]/64). \quad (2.6)$$

Na predição Inter do tipo B, a estimação de movimento é feita com base nos quadros armazenados nas listas 0 e 1, com a possibilidade de realizar a bipredição. Na bipredição, o bloco de referência é obtido através de uma média entre um bloco da lista 0 e outro da lista 1. A equação a seguir define os valores dos pixeis de referência no processo de bipredição, onde $pred[x, y]$ é o valor da amostra a ser predita; $predL0[x, y]$ e $predL1[x, y]$ são os valores das amostras provenientes do bloco da lista 0 e 1 respectivamente.

$$pred[x, y] = (predL0[x, y] + predL1[x, y] + 1) >> 1. \quad (2.7)$$

Note que a operação $x >> y$ é o deslocamento de y bits para a direita da representação binária do número x. E portanto, a operação $(x + 1) >> 1$ equivale

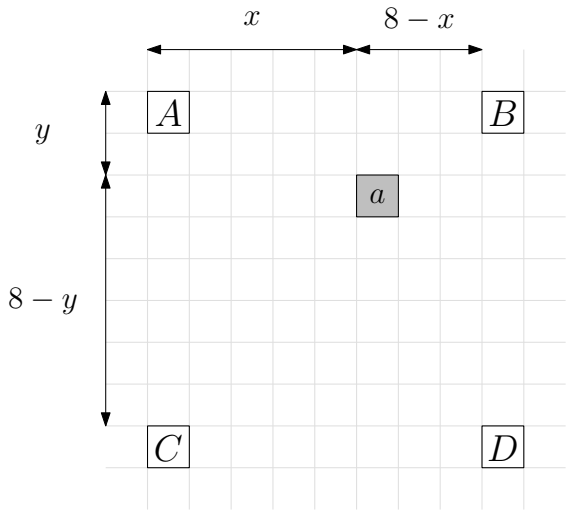


Figura 2.4: Interpolação para as componentes de croma

a $round(x/2)$[1]. Pode-se ainda utilizar uma bipredição ponderada, onde se aplicam os fatores multiplicativos w_0 e w_1 a cada uma das referências.

Os macroblocos das predições Inter podem ser divididos em partições de 16×16, 8×16, 16×8 e 8×8 pixeis. Se o macrobloco for dividido em 4 partições 8×8, cada partição 8×8 poderá ainda ser dividida em subpartições de 8×8, 8×4, 4×8 e 4×4 pixeis, chamados de submacroblocos. Para a predição Inter do tipo B, o tipo de predição – lista 0, lista 1 ou bipredição – para partições de 16×16 a 8×8 é escolhido de forma independente. Entretanto, o mesmo tipo de predição é utilizado em todos os submacroblocos dentro de uma mesma partição 8×8.

O quadro utilizado como referência na estimação de movimento também é escolhido de forma independente para blocos de 16×16 a 8×8. Porém, apenas um quadro é utilizado para todas as subpartições dentro de uma partição 8×8. O mesmo ocorrerá no caso de se usar a bipredição.

Para que o decodificador obtenha o mesmo bloco de referência é preciso que o mesmo saiba a posição exata do bloco. Deste modo, é preciso que sejam codificados a lista, o quadro e o vetor de movimento relativo entre a posição do bloco original e o de referência com precisão de até 1/4 de pixel. Entretanto, como blocos adjacentes tendem a ter vetores de movimento com valores próximos, pode-se ainda predizer o vetor de movimento e somente codificar a diferença entre este e sua predição.

Depois de obtido o bloco de referência é então calculada a diferença entre o mesmo e o bloco original, ou seja, o resíduo. Os resíduos são transformados empregando a transformada discreta do cosseno (*Discrete Cosine Transform* - DCT) de tamanho 4×4 ou 8×8. Os coeficientes DC's das DCT's são ainda transformados através da transformada de Hadamard para aproveitar ainda mais a redundância

[1]Geralmente na norma do H.264/AVC, as operações de arredondamento e quantização são calculadas apenas utilizando adição e deslocamentos binários.

entre blocos adjacentes. Os coeficientes são então quantizados e codificados por um codificador aritmético juntamente com as diferenças entre vetores de movimento, o tipo de particionamento e o quadro de referência.

2.3 Codificação por Entropia

A codificação por entropia da informação de movimento no padrão H.264/AVC pode ser feita de duas maneiras – a codificação de tamanho variável com contextos adaptativos (*Context-based Adaptive Variable Length Coding* - CAVLC) ou a codificação binária aritmética com contextos adaptativos (*Context-based Adaptive Binary Arithmetic Coding* - CABAC).

Para o CABAC, cada elemento de sintaxe[2] (*Syntax Element* - SE) da informação de movimento é binarizado de acordo com algum método especificado pela norma H.264/AVC [1] [7]. Como exemplo, podemos citar:

- Tabelas fixas [1] para listas e tipo de particionamento;
- Binarização unária [1] para índices das referências;
- Concatenação das binarizações unária truncada e binarização de terceira ordem de Exp-Golomb [1] para vetores de movimento.

Cada bit da binarização é chamado de bin, e serão esses bins que serão codificados utilizando um determinado contexto. Cada contexto é inicializado com uma das 63 possíveis probabilidades do símbolo menos provável (0 ou 1) e a indicação do símbolo menos provável [1], os quais são utilizados pelo codificador aritmético. O codificador atualiza o contexto apropriadamente de acordo com o símbolo codificado anteriormente [1] [7]. A norma H.264/AVC define a quantidade de contextos para cada bin, assim como a inicialização de cada contexto [1]. Quando o bin tem a possibilidade de ser codificado em mais de um contexto, a escolha do contexto é feita de acordo com as informações de movimento de blocos adjacentes já processados [1].

Não é objetivo desta dissertação explicar detalhadamente o funcionamento do CABAC, assim como da norma H.264/AVC. Todas as informações a respeito do padrão H.264/AVC podem ser encontradas com mais detalhes na própria norma [1]. Nesse capítulo, foi apresentado apenas o básico sobre a norma H.264/AVC para a criação de um codificador de vídeo sem transformadas. No próximo capítulo, será criado um codificador de vídeo com base no casamento de padrões utilizando apenas ferramentas fornecidas pelo padrão H.264/AVC.

[2]Elemento de sintaxe (SE) é todo elemento de dados presentes no *bitstream* do padrão H.264/AVC [1].

Capítulo 3

Casamento de Padrões Compatível com a Norma H.264/AVC

A primeira abordagem para observar o efeito do casamento de padrões em vídeo, descartando assim as transformadas, foi criar um codificador de vídeo que descartasse todos os resíduos. Deste modo, esse novo codificador irá utilizar a estimação de movimento para realizar a operação de casamento de padrões e os quadros codificados anteriormente seriam o "dicionário adaptativo". As palavras desse dicionário seriam os vetores de movimento, as listas e referências.

3.1 Descarte de Resíduos em *Slices* B – DRSB

Como descrita na Seção 2.2, a estimação de movimento realizada em *slices* do tipo B é feita com base mos quadros das listas 0 e 1, além da bipredição. Assim, pode-se considerar que a estimação de movimento é mais refinada nos *slices* do tipo B [8]. Portanto, é de se esperar que ao utilizar a compensação de movimento descartando os resíduos obteremos uma melhor aproximação do bloco original nesses *slices* do que nos *slices* do tipo P. O novo codificador irá descartar os resíduos somente em *slices* do tipo B, deixando os demais *slices* iguais aos *slices* do codificador H.264/AVC. Assim, uma forma de manter o desempenho é realizar a estimação de movimento com base nos quadros codificados somente com *slices* I ou P, já que nestes *slices* não será feito o descarte de resíduos.

Para implementar o novo codificador foi utilizado o software de referência do padrão H.264/AVC, versão JM 15.0 (16/12/2008) [9]. A modificação feita no programa original é simplesmente zerar todos os coeficientes residuais antes do processo de codificação [10]. O *Coded Block Pattern* (CBP) é o SE responsável por informar ao decodificador quais blocos do macrobloco possuem resíduos codificados e, no caso de *slices* do tipo B, o CBP será sempre zero no novo codificador. Desse modo, o

Tabela 3.1: Valores de QP utilizados.

Tipo de *Slice*	QP								
B	45	40	35	30	25	20	15	10	05
I e P	43	38	33	28	23	18	13	08	03

decodificador será informado da existência dos blocos de resíduos nulos e não entrará na rotina de decodificação de resíduos.

Todas as simulações envolveram a codificação dos primeiros 99 quadros das sequências de teste *Akiyo* (resolução QCIF), *Foreman* (CIF), *Flower Garden* (SIF) e *Mobile & Calendar* (CIF), tendo as componentes de croma sido subamostradas segundo o padrão 4:2:0. Essas sequências foram escolhidas por serem largamente usadas como referências em testes de codificação de vídeo [8] e também por abrangerem diversos tipos de sequências de vídeos: desde apenas uma pessoa falando até sequências com grande quantidade de movimento.

A configuração utilizada incluiu:

- Perfil *High*
- Codificação bipreditiva
- Transformadas 4×4 e 8×8
- 1 *slice* por quadro
- *Slices* Intra: somente o primeiro
- Lista 0: 5 quadros de referência (somente *slices* I ou P)
- Lista 1: 1 quadro de referência (somente *slices* P)
- GOP (*Group Of Pictures*): IBPBPBPBP...
- Modo Intra habilitado para blocos de *slices* Inter
- Otimização taxa-distorção com complexidade alta

Um teste adicional foi realizado para a sequência *Foreman*, com a definição do GOP alterada para o seguinte padrão: IBBBPBBBP...

Como o controle de taxa do software de referência ainda é deficiente, foi escolhido utilizar codificação por taxa variável através de diferentes passos de quantização de resíduos. Assim, diferentes taxas – em bits/quadro ou kb/s – podem ser alcançadas variando o parâmetro de quantização (*Quantization Parameter* - QP) para cada tipo de *slice*. Os valores utilizados nas simulações aqui descritas são apresentados

na Tabela 3.1[1]. É interessante mencionar que há uma relação direta entre os valores de QP e o multiplicador de Lagrange λ utilizado no cálculo do custo taxa-distorção utilizado pelo software de referência [9]. Assim, para cada QP o codificador irá escolher o modo de codificação que forneça o menor custo a partir do operador de Lagrange $D + \lambda R$, onde D é a distorção e R é a taxa. A qualidade dos quadros ou distorção dos quadros é medida através da escala logarítmica da razão sinal (quadro original) para o ruído (quadro codificado) (*Peak Signal to Noise Ratio* - PSNR) [6]. A medida de distorção PSNR em decibéis é definida da seguinte forma:

$$\mathrm{PSNR}_{dB} = 10 log_{10} \frac{255^2}{\mathrm{MSE}}. \tag{3.1}$$

MSE (*Mean Squared Error*) é o erro médio quadrático [6]. Se considerarmos quadros com resolução $M \times N$, o MSE será definido como:

$$\mathrm{MSE} = \frac{1}{N \cdot M} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (o_{i,j} - r_{i,j})^2, \tag{3.2}$$

onde $o_{i,j}$ e $r_{i,j}$ representam os valores dos pixeis dos quadros originais e reconstruídos respectivamente na posição i e j. Resumindo, variando o valor de QP é possível controlar o valor de λ e, portanto, a taxa [8].

Em todas as simulações aqui realizadas, o comportamento das curvas taxa-distorção para as componentes de croma (U e V) foi muito similar ao comportamento das curvas da componente luma (Y'), como pode ser observado nas Figuras 3.1, 3.3 e 3.4, onde somente dados de *slices* do tipo B são apresentados. Como os *slices* do tipo I e P são os mesmos do codificador original, acrescentá-los nos gráficos apenas dificulta a análise do efeito do descarte de resíduos. A comparação entre as Figuras 3.1 e 3.5, na qual os dados de todos os *slices* estão presentes, ilustra bem esse fato. Desse modo, a fim de permitir uma melhor visualização e evitar o excesso de curvas, serão apresentados somente os resultados para componente luma dos *slices* do tipo B nas figuras seguintes. Porém, no Apêndice A.1 é possível visualizar as figuras para as componentes de croma de todas as sequências (*Akiyo, Foreman, Flower Garden e Mobile & Calendar*), além das figuras contendo os dados de todos os tipos de *slices* (I, B e P).

As sequências, *Foreman*, *Mobile & Calendar* e *Flower Garden* possuem grande quantidade de movimento, inclusive com mudanças de fundo (*background*). Nestes casos, conforme apresentado nas Figuras 3.1, 3.6 e 3.7, o codificador DRSB possui

[1]Nem todos os valores de QP, presentes na Tabela 3.1, foram utilizados ao plotar as curvas taxa-distorção contidas nas figuras deste capítulo. Para uma melhor visualização, os valores para QP ≤ 10 foram utilizados apenas nas curvas taxa-distorção da sequência *Akiyo*.

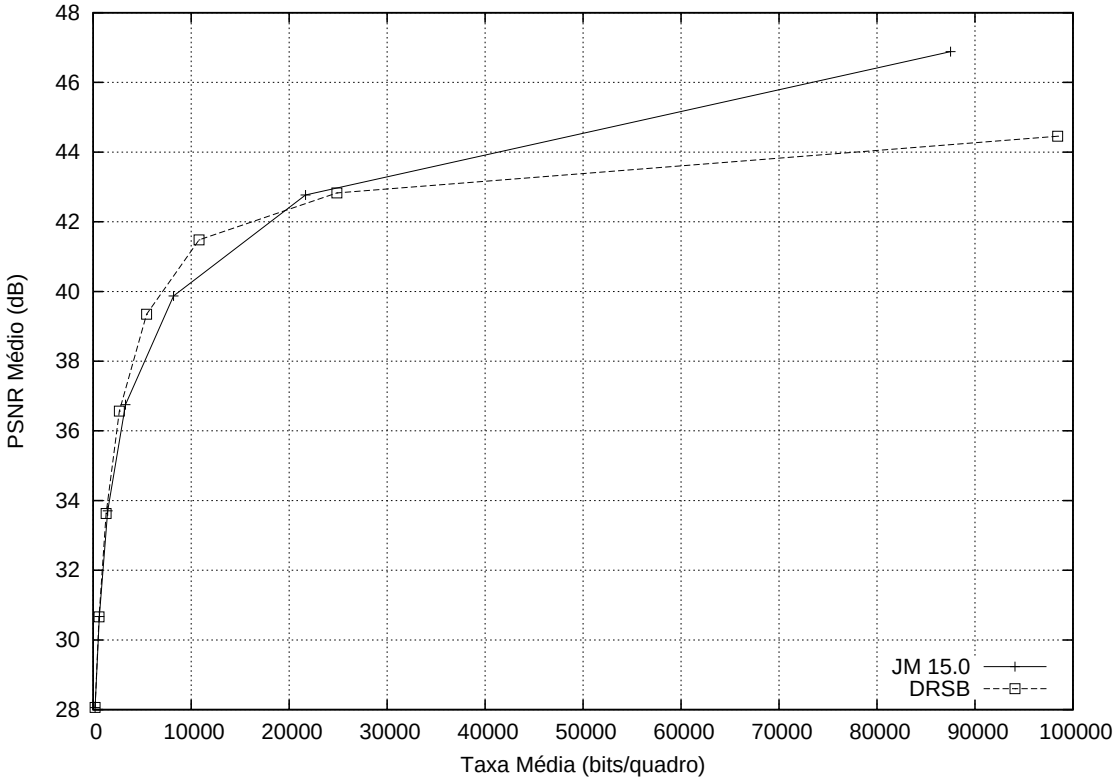
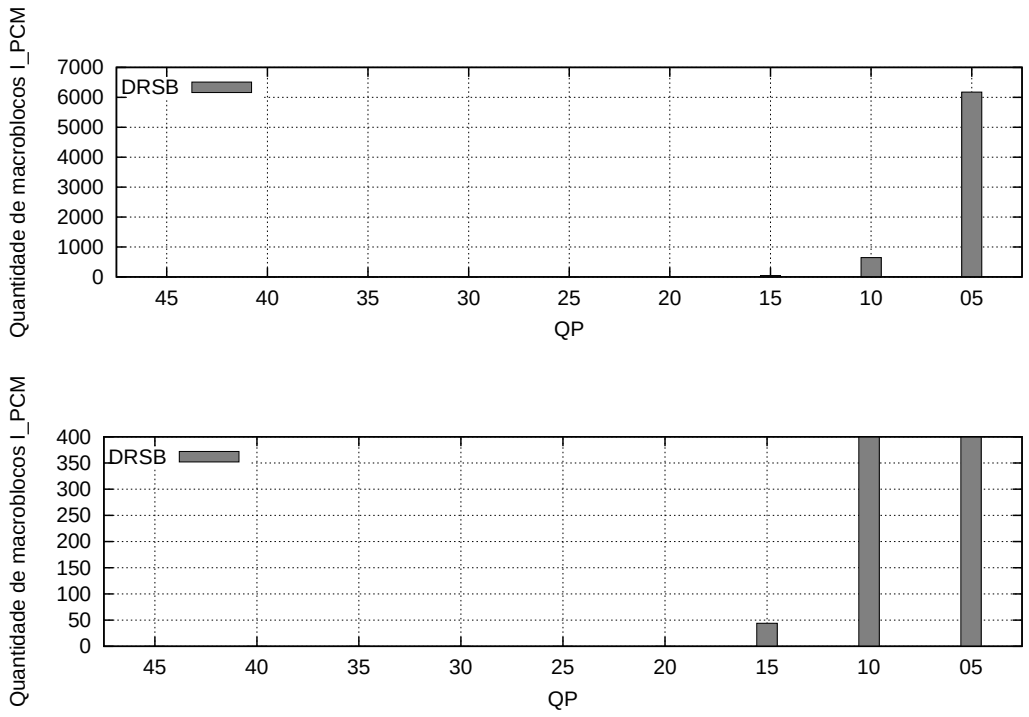


Figura 3.1: *Foreman* (CIF) - Somente quadros com *slices* do tipo B (Luma)



(b) Zoom

Figura 3.2: Quantidade de macroblocos I_PCM codificados na sequência *Foreman* (CIF) em *slices* do tipo B

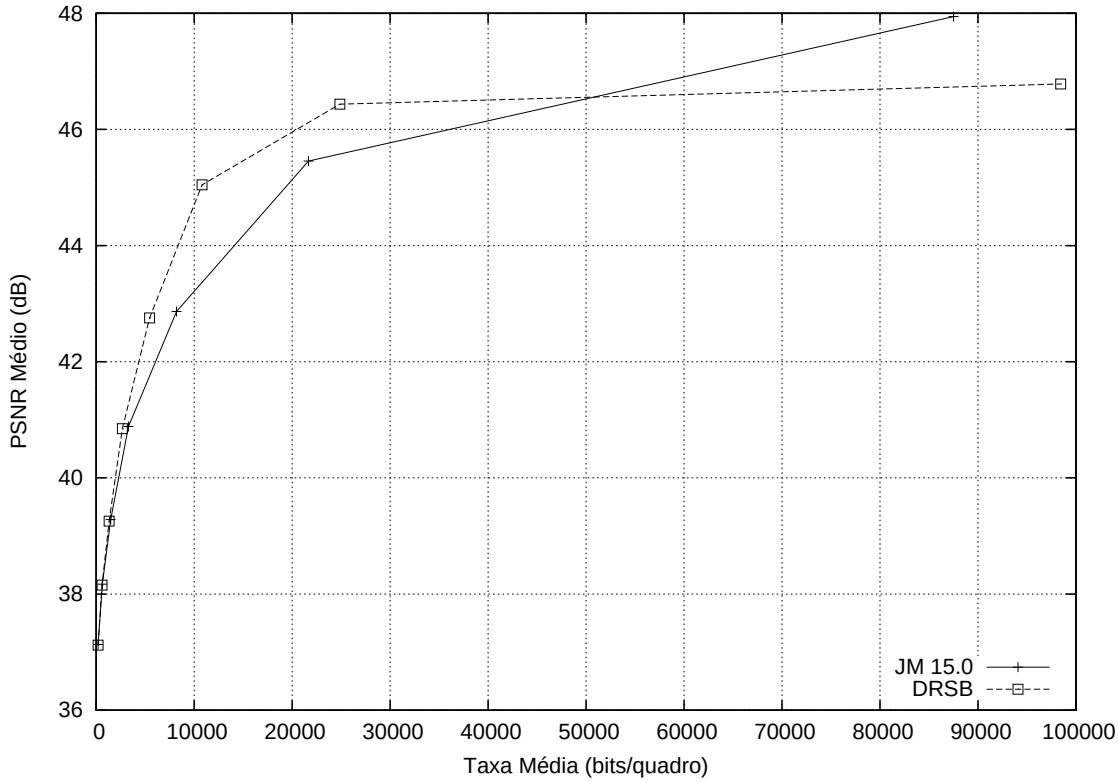


Figura 3.3: *Foreman* (CIF) - Somente quadros com *slices* do tipo B (Croma - U)

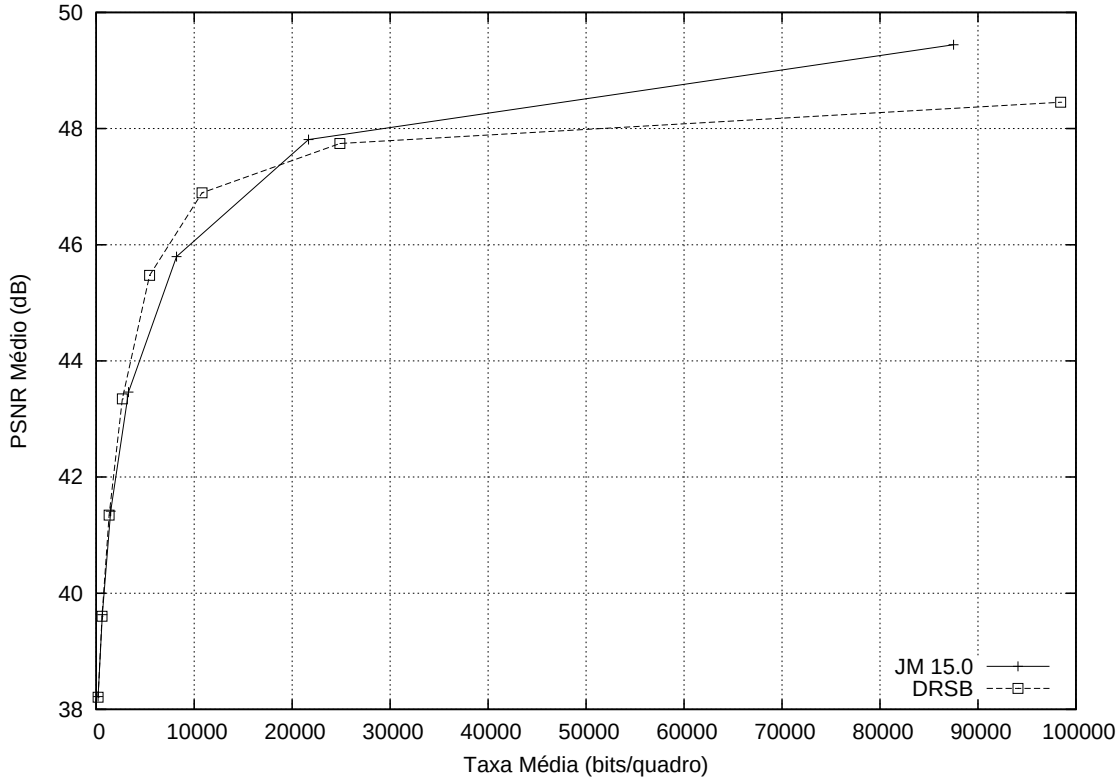


Figura 3.4: *Foreman* (CIF) - Somente quadros com *slices* do tipo B (Croma - V)

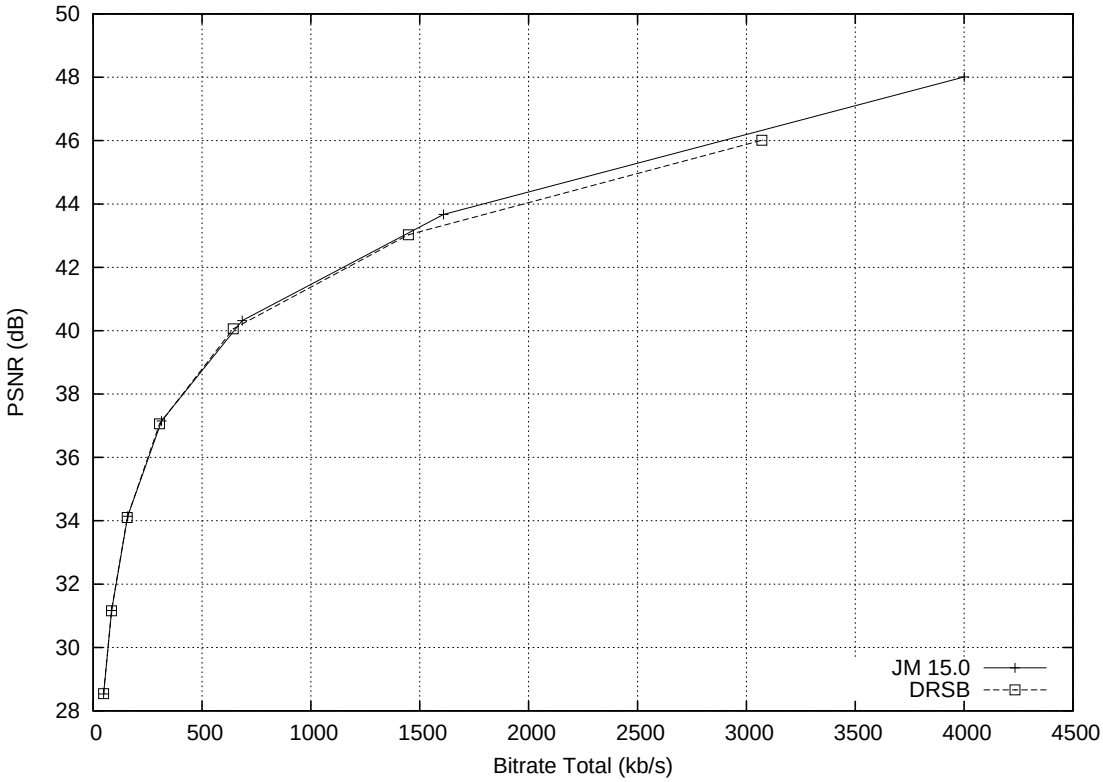


Figura 3.5: *Foreman* (CIF) - Todos os quadros (Luma)

melhor desempenho para baixas e médias taxas[2], onde os valores de QP e λ são altos. Logo, nessas taxas, o custo está relacionado mais com a taxa do que com a distorção. Em outras palavras, para médias e baixas taxas, somente a estimação de movimento é suficiente para obter a mesma distorção do codificador original com um custo mais baixo. À medida que os valores de QP e λ diminuem (QP < 25), o valor da distorção tem maior influência no valor do custo. Nesses casos, mesmo aumentando o número de blocos 4×4, o codificador DRSB não consegue alcançar um custo tão baixo quanto o codificador original. Dessa maneira, a única forma de diminuir a distorção, e consequentemente o custo, é aumentando o número de macroblocos codificados com o I_PCM. No modo I_PCM, como descrito na Seção 2.1, o valor de cada pixel (luma e croma) é transmitido sem codificação, penalizando a taxa para obter uma distorção nula e criando, assim, um limite superior para o custo. O elevado número de macroblocos I_PCM é o principal responsável pelo aumento significativo da taxa no método DRSB, limitando assim o desempenho do mesmo para altas taxas. Na Figura 3.2, é possível notar o alto número de macroblocos codificados com a codificação I_PCM para baixos valores de QP.

[2]Como as sequências simuladas possuem diferentes resoluções (QCIF, CIF e SIF), define-se as taxas alta, média e baixa como diferentes valores de QP e não os valores reais obtidos em bits/quadro. Por exemplo, baixa taxa significa valores de QPs altos, próximos de 40, podendo a mesma ser igual a 200 ou 10000 bits/quadro. Alta taxa são as taxas obtidas com valores de QP menores ou iguais a 15.

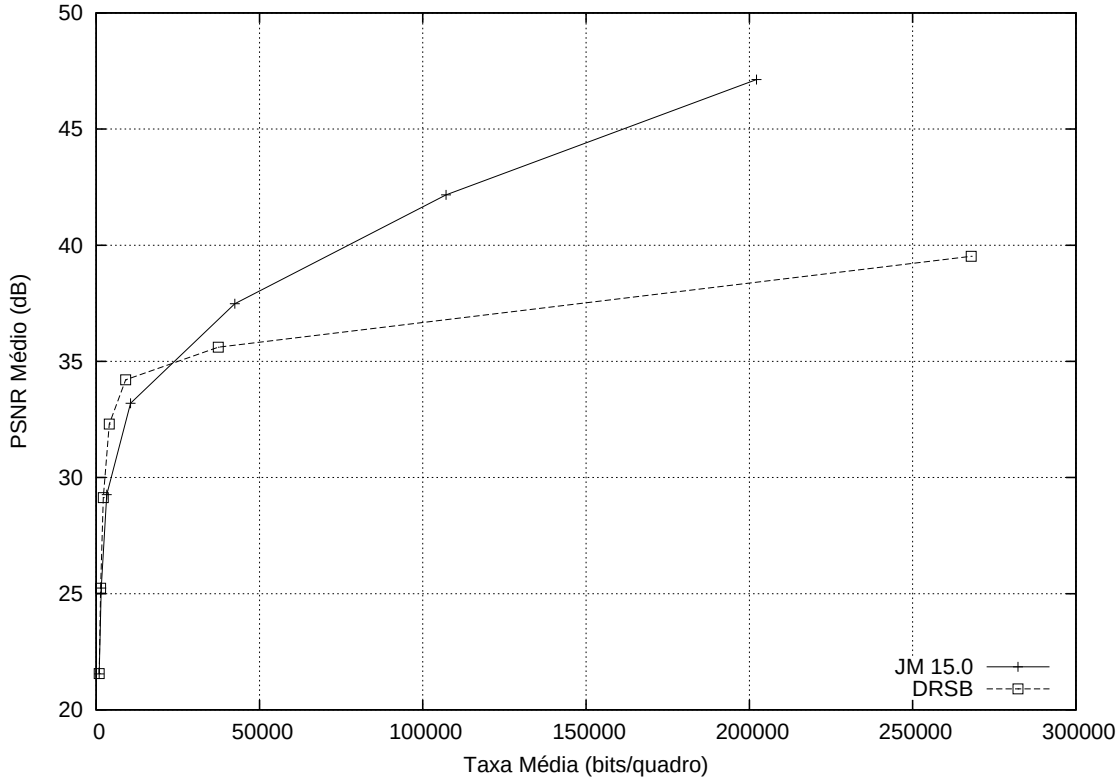


Figura 3.6: *Flower Garden* (SIF) - Somente quadros com *slices* do tipo B (Luma)

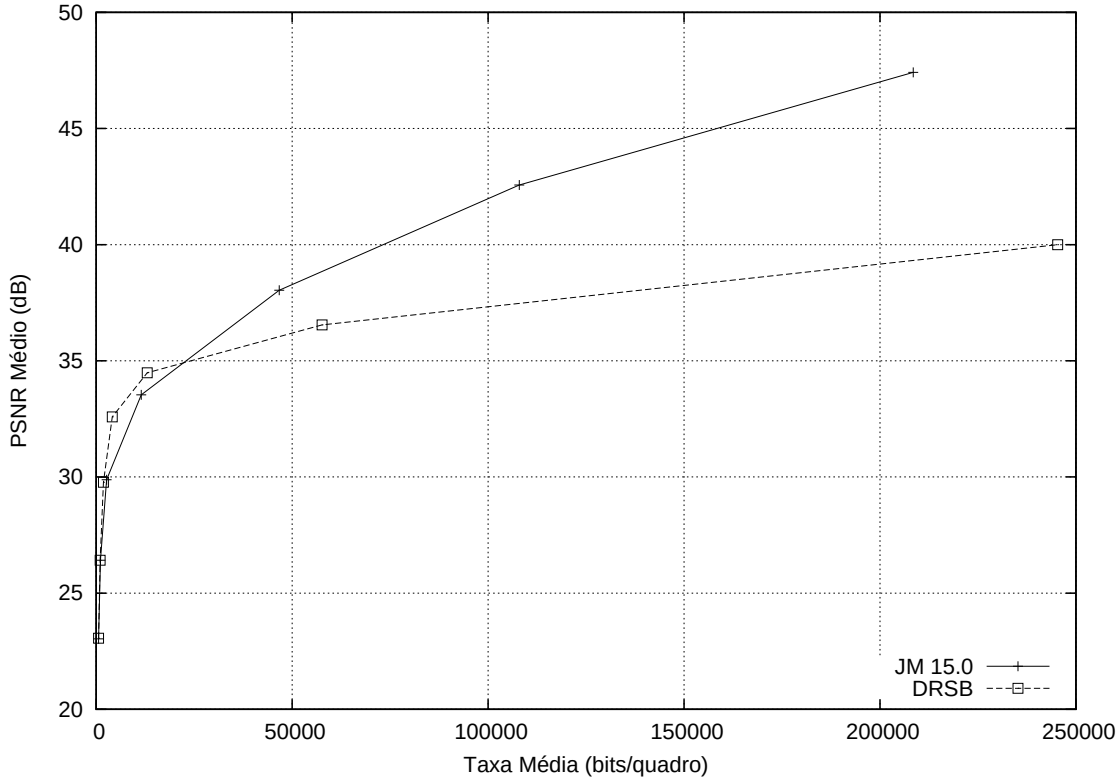


Figura 3.7: *Mobile & Calendar* (CIF) - Somente quadros com *slices* do tipo B (Luma)

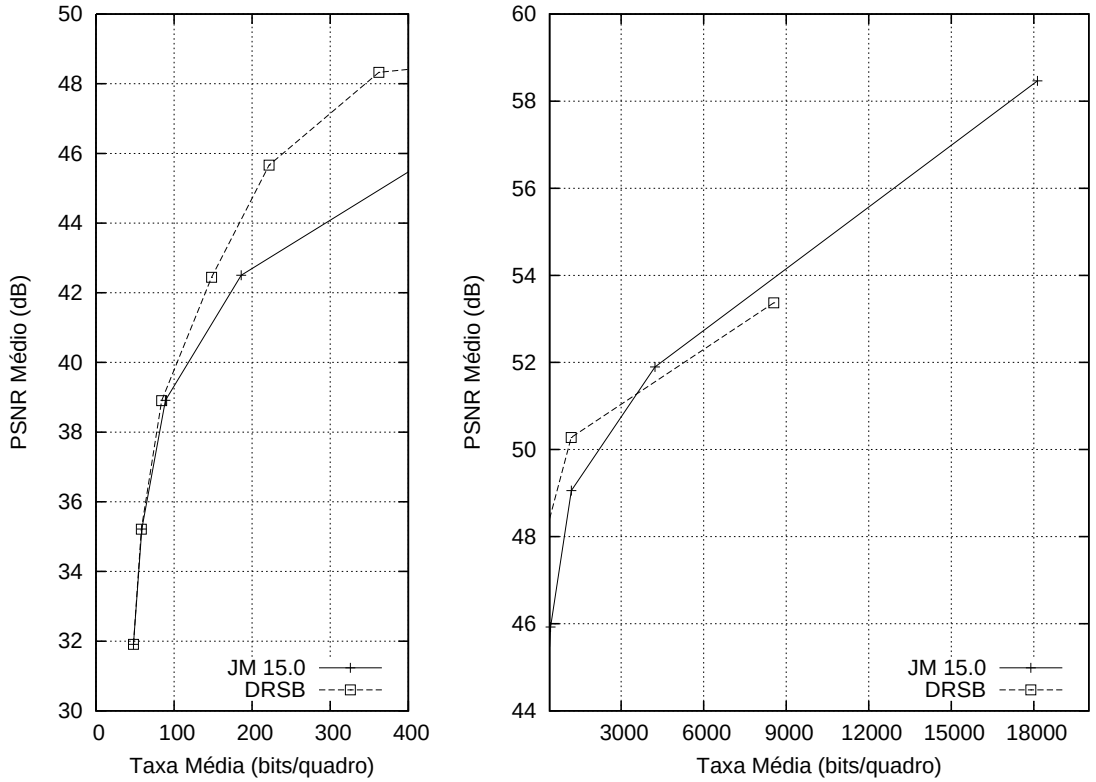


Figura 3.8: *Akiyo* (QCIF) - Somente quadros com *slices* do tipo B (Luma)

A próxima sequência a ser analisada – *Akiyo*, com resolução QCIF – diferencia-se das demais por ter quadros de pequena dimensão (176×144 pixeis) e pouco movimento dos objetos apresentados. Em consequência, o codificador DRSB, onde não há envio dos dados referentes à transformada dos resíduos, possui melhor desempenho em relação ao JM 15.0 em baixa a média taxas (QP alto). Somente com um QP bastante pequeno (QP $\leq$ 10) – taxas muito altas – é que o JM 15.0 possui um desempenho melhor. Contudo, conforme pode ser analisado na Figura 3.8, em taxas médias e baixas de codificação, apesar da quantidade de bits utilizada ter sido significativamente menor no DRSB, a PSNR média para quadros de *slices* do tipo B é bastante similar. Por exemplo, para QP = 20 a Taxa Média[3] (221,59 bits/quadro) do codificador DRSB é quase metade da Taxa Média (432,90 bits/quadro) do codificador JM 15.0, apesar disto a diferença das PSNRs é de apenas 0,258 dB.

Um último teste foi executado para estudar o efeito do GOP no desempenho dos codificadores. Na comparação entre as Figuras 3.9 e 3.1 verifica-se que utilizando um GOP com múltiplos quadros com *slices* do tipo B consecutivos (IBBBP), tanto o método aqui proposto quanto o JM 15.0 apresentam resultado semelhante ao re-

[3]Taxa Média (com letras T e M maiúsculas) é a quantidade média de bits para a codificação de um quadro utilizando um determinado QP, com unidade bits/quadro. Diferenciando-se de taxa média (com letras t e m minúsculas) que indica codificação utilizando QP com valor próximo de 25.

sultado com o GOP original (IBP). Uma sugestão de trabalho futuro é melhorar a configuração utilizada para os codificadores DRSB e H.264/AVC, a modificação do GOP foi apenas um primeiro passo nessa direção. Apesar deste trabalho reportar resultados apenas para uma sequência, foi observado que o desempenho dos codificadores tende a independer do tipo de GOP para a configuração utilizada.

Nesta seção, ao estudar o efeito do casamento de padrões em vídeo utilizando ferramentas do H.264/AVC obteve-se um desempenho melhor do que o codificador original para médias e baixas taxas de uma maneira em geral. A ideia da próxima seção é criar uma forma de aproveitar o ganho obtido com o descarte de resíduos nos decodificadores H.264/AVC atuais.

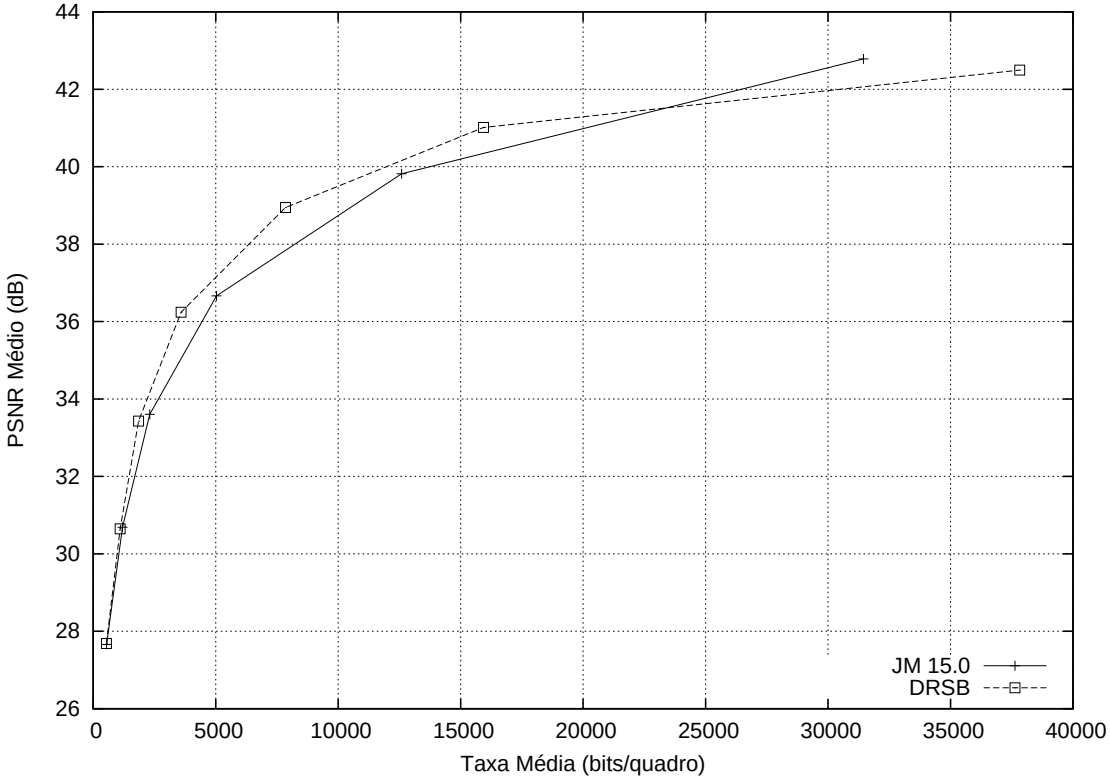


Figura 3.9: *Foreman* (CIF) - Somente quadros com *slices* do tipo B com GOP IBBBP (Luma)

3.2 Otimização na Codificação de *Slices* B – OCSB

A análise dos resultados obtidos com o método DRSB apresentado na Seção 3.1, indica uma melhora de desempenho de codificação para determinadas taxas quando não são transmitidos os resíduos dos *slices* do tipo B . Assim, a inclusão de um método de otimização taxa-distorção que leve em conta a possibilidade de igualar

a zero ou não os coeficientes do resíduo de predição pode resultar em ganhos ainda maiores de desempenho [11].

O método proposto nesta seção altera o laço de otimização taxa-distorção para quadros com *slices* do tipo B. O codificador OCSB inclui modos de descarte dos coeficientes da transformada do resíduo de predição para cada um dos modos de particionamento do macrobloco que está sendo codificado. Vale ressaltar que o objetivo desta seção não é criar um codificador sem transformadas, mas adequar o ganho de desempenho obtido com o DRSB para os decodificadores atuais do padrão H.264/AVC.

O laço de otimização taxa-distorção é alterado da seguinte forma: o custo de codificação $D + \lambda R$ é calculado para cada um dos possíveis modos, com e sem a exclusão dos resíduos, sendo aquele de menor custo o escolhido pelo codificador. Dessa forma esse novo método combina o desempenho do codificador DRSB para baixas taxas com o do codificador JM 15.0 para altas taxas.

O conjunto de sequências utilizado foi o mesmo da Seção 3.1 (*Akiyo*, *Foreman*, *Mobile & Calendar* e *Flower Garden*). O valores de QP utilizados também foram os mesmos da seção anterior, como mostrado na Tabela 3.1. Para uma melhor comparação, a mesma configuração dos codificadores foi utilizada:

- Perfil *High*
- Codificação bipreditiva
- Transformadas 4×4 e 8×8
- 1 *slice* por quadro
- *Slices* Intra: somente o primeiro
- Lista 0: 5 quadros de referência (somente *slices* I ou P)
- Lista 1: 1 quadro de referência (somente *slices* P)
- GOP (*Group Of Pictures*): IBPBPBPBP...
- Modo Intra habilitado para blocos de *slices* Inter
- Otimização taxa-distorção com complexidade alta

O mesmo teste para a sequência *Foreman* com GOP alterado foi executado para verificar o desempenho do novo codificador ao incluir mais *slices* do tipo B. Alterando-se o GOP para o seguinte padrão: IBBBPBBBP...

Novamente, pelo mesmo motivo, serão somente mostrados somente os gráficos referentes à componente luma (Y') das sequências em quadros com *slices* do tipo

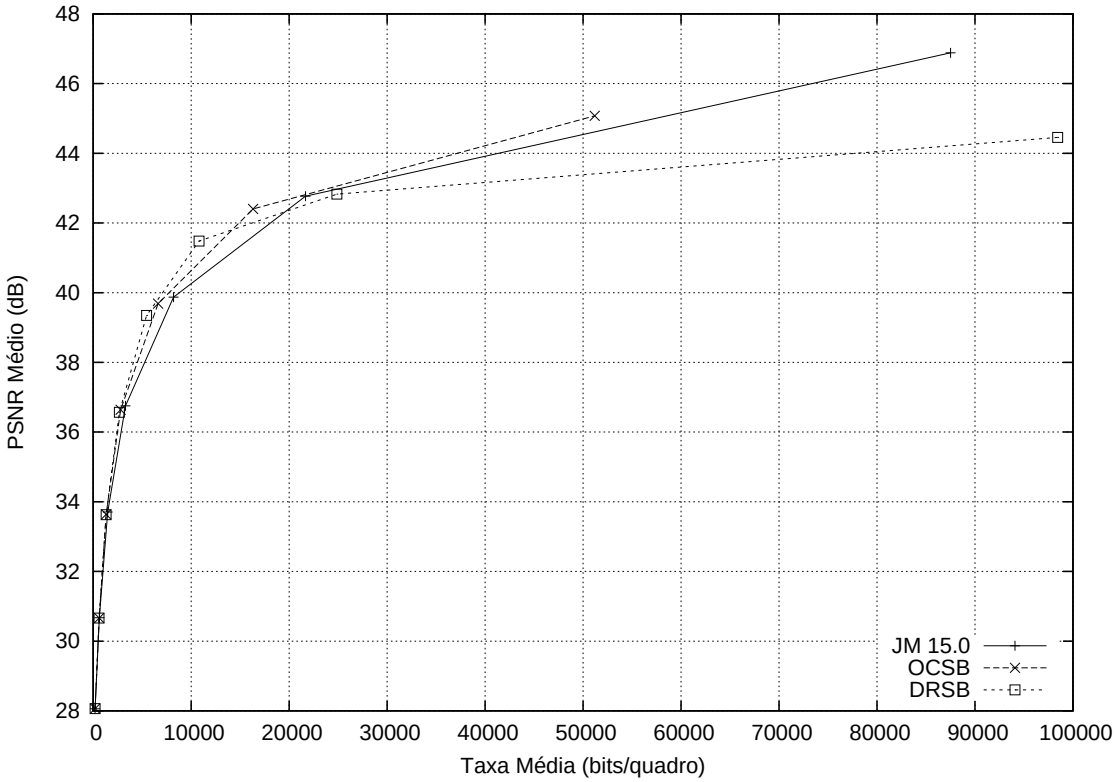


Figura 3.10: *Foreman* (CIF) - Somente quadros com *slices* do tipo B (Luma)

B. Porém, outros gráficos contendo todos os *slices* com componentes de luma ou croma podem ser observados no Apêndice A.2, juntamente com os de componentes de croma somente para quadros com *slices* do tipo B.

Para as sequências *Foreman*, *Mobile & Calendar* e *Flower Garden*, como pode ser observado nas Figuras 3.10, 3.11 e 3.12, o método OCSB produz melhores resultados em taxas baixas e altas. Em taxas baixas, o codificador OCSB descarta a maioria dos resíduos dos *slices* do tipo B, obtendo praticamente o mesmo desempenho do codificador DRSB. Pois nessas taxas os valores de QP e λ são altos, fazendo com que a taxa R tenha um peso maior do que a distorção D no cálculo do custo ($D + \lambda R$). Para taxas altas (valores baixos de QP e λ), o codificador OCSB evita o alto custo devido à escolha do modo I_PCM utilizando os resíduos e as transformadas 8×8 ou 4×4. Dessa forma, em altas taxas, o codificador OCSB obtém um desempenho igual ou melhor do que codificador original. Para taxa média (QP próximo de 25), é possível notar que o desempenho do OCSB supera tanto o JM 15.0 quanto o DRSB. Por exemplo, para a sequência *Flower Garden* onde para taxas média e alta (QP $\leq$ 25 ou Taxa Média $>$ 15000 bits/quadro), o desempenho do OCSB supera os codificadores JM 15.0 e DRSB, como pode ser observado na Figura 3.11.

A sequência *Akiyo*, como descrita anteriormente, é uma sequência de pouco movimento e baixa resolução (QCIF). Assim, ela favorece o método DRSB, o qual apresenta melhor relação custo-benefício dentre os três codificadores para essa

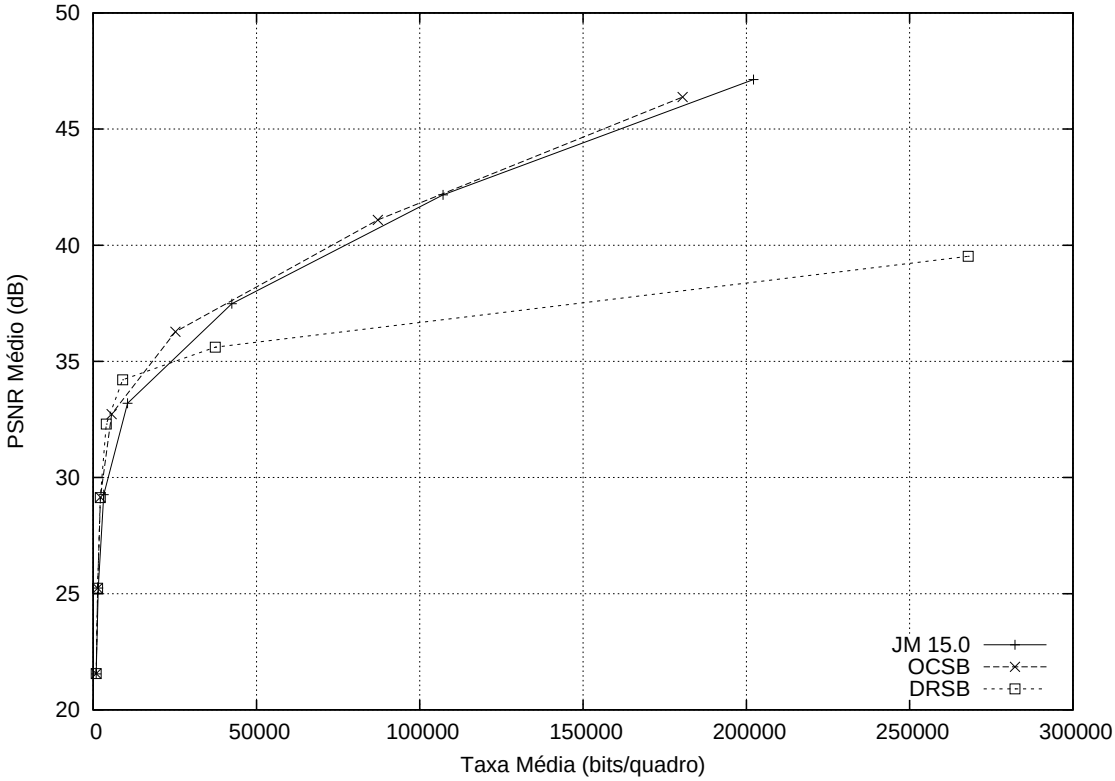


Figura 3.11: *Flower Garden* (SIF) - Somente quadros com *slices* do tipo B (Luma)

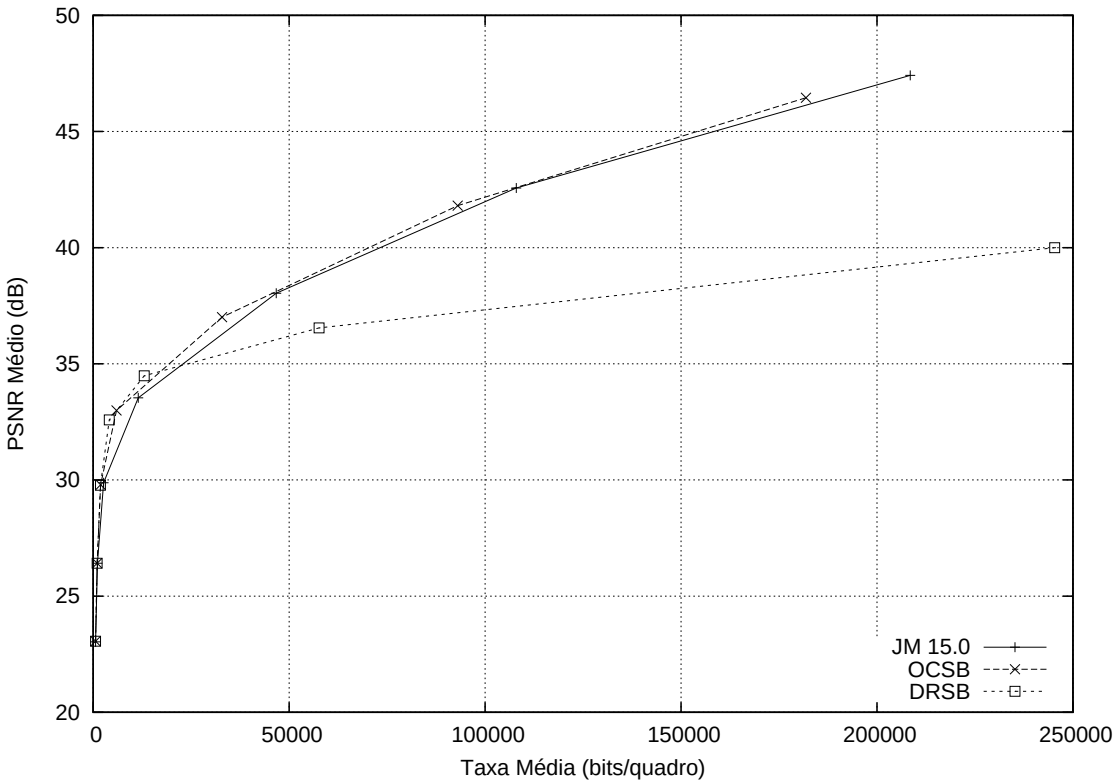


Figura 3.12: *Mobile & Calendar* (CIF) - Somente quadros com *slices* do tipo B (Luma)

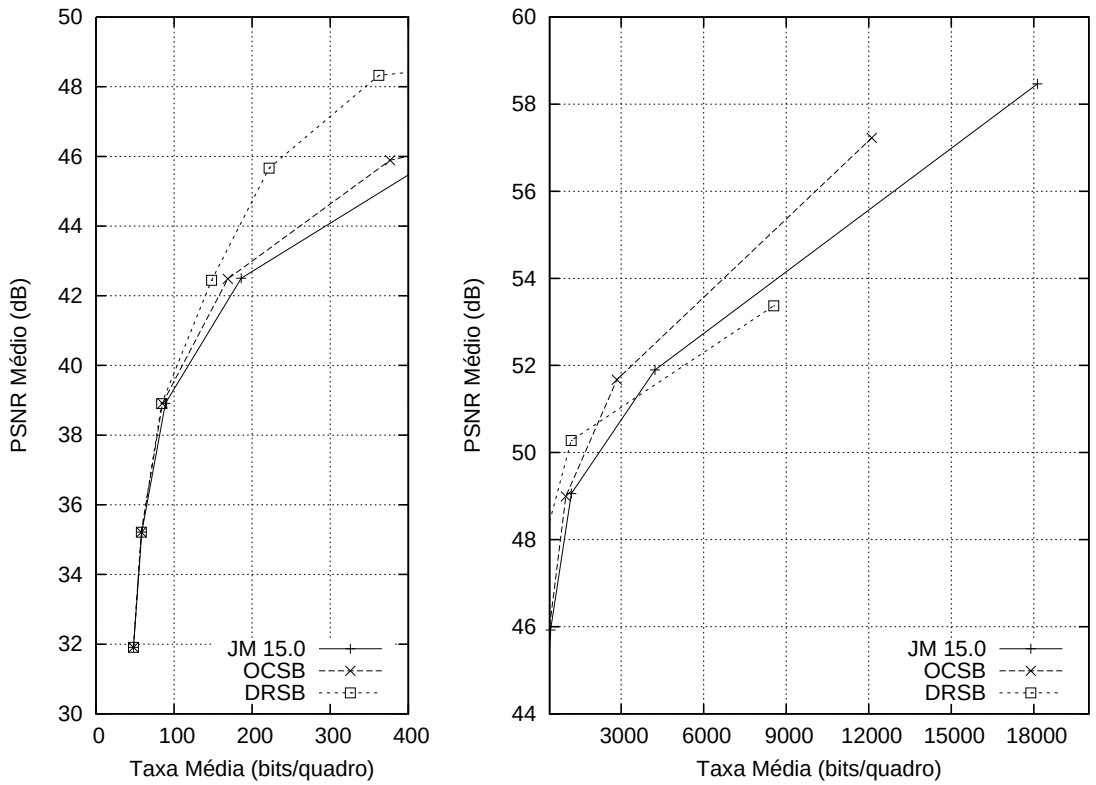


Figura 3.13: *Akiyo* (QCIF) - Somente quadros com *slices* do tipo B (Luma)

sequência, em baixa e média taxas. Contudo, para altas taxas, (QP $\leq$ 10 ou Taxa Média $>$ 2000 bits/quadro) o desempenho do codificador OCSB supera o desempenho dos codificadores JM 15.0 e DRSB. Os resultados com todos os codificadores para esta sequência encontram-se na Figura 3.13.

Assim como na Seção 3.1, o aumento na quantidade de *slices* do tipo B no GOP não altera o desempenho dos codificadores. Utilizando o GOP (IBBBP) o desempenho dos codificadores foi basicamente o mesmo em relação ao GOP anterior (IBP), como pode ser observado nas Figuras 3.10 e 3.14.

É importante notar que tais modificações no sistema (DRSB e OCSB) não contrariam as especificações do padrão H.264/AVC – que tratam unicamente da sintaxe da *bitstream* a ser decodificada – sendo modificações que afetam exclusivamente o comportamento do codificador. Portanto, os métodos aqui propostos são totalmente compatíveis com o padrão atual. Outro ponto é que o codificador OCSB possui sempre um desempenho melhor ou igual ao codificador H.264/AVC.

Neste capítulo, foi apresentado um codificador de vídeo baseado em casamento de padrões utilizando somente ferramentas do H.264/AVC. Na Seção 3.2, foi apresentada uma maneira de otimizar o codificador H.264/AVC sem a necessidade de alterar os decodificadores H.264/AVC. No próximo capítulo, será criada uma implementação de um codificador de vídeo com base inteiramente no casamento de padrões, utilizando algumas ferramentas do H.264/AVC, mas sem utilizar transformadas.

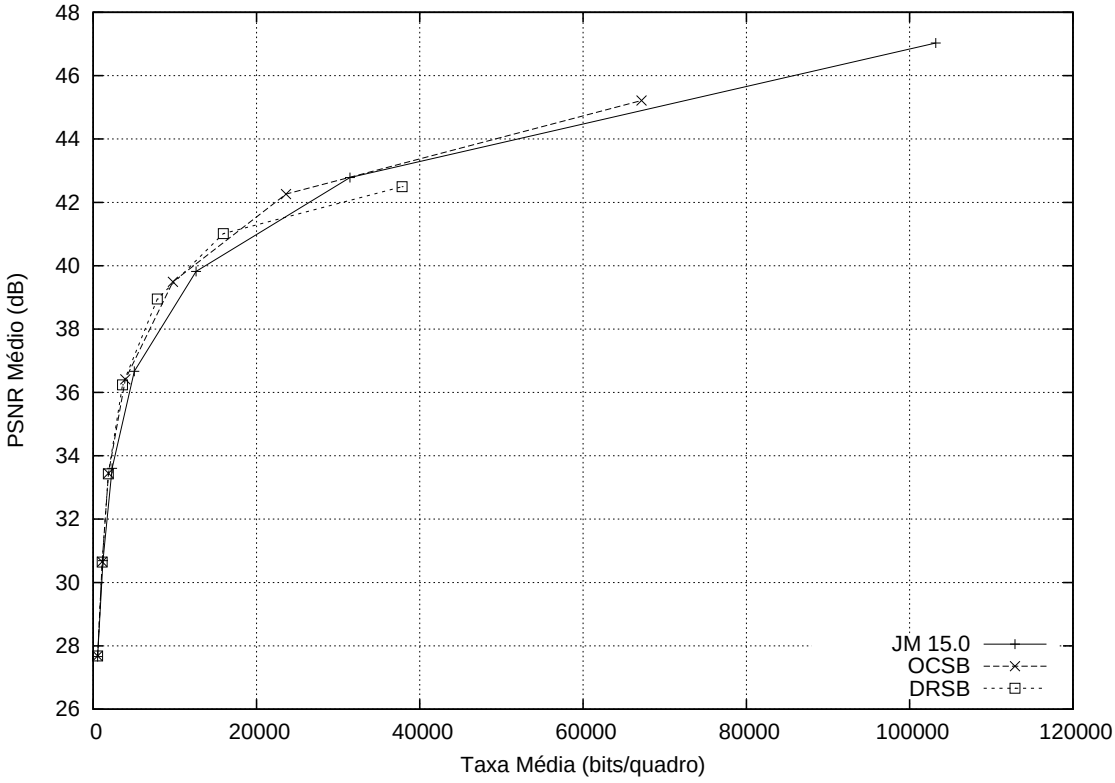


Figura 3.14: *Foreman* (CIF) - Somente quadros com *slices* do tipo B com GOP IBBBP (Luma)

Capítulo 4

Uma Modificação Incompatível com a Norma H.264/AVC Utilizando Casamento de Padrões

Neste capítulo, será apresentado um novo codificador de vídeo, gerado a partir do padrão H.264/AVC, cuja principal característica é usar uma técnica de casamento de padrões e sem utilizar transformadas – CPST. Nas seções seguintes, será apresentada a implementação desse codificador, assim como os resultados experimentais.

A análise dos resultados do método DRSB descrito na Seção 3.1 demonstra a vantagem de utilizar o casamento de padrões para codificação de vídeo em *slices* do tipo B: melhor desempenho para baixas e médias taxas. Porém, para altas taxas, o codificador fica limitado em não diminuir a distorção ao descartar o resíduo da compensação de movimento. Deste modo, acaba por penalizar a taxa empregando a codificação de macroblocos I_PCM, como descrito anteriormente. O objetivo do novo codificador é aproveitar esse bom desempenho do codificador DRSB em baixas e médias taxas e melhorá-lo para altas taxas. Com a diferença que serão codificados utilizando casamento de padrões tanto os *slices* do tipo B como os do tipo P. Uma forma de melhorar a qualidade em altas taxas seria refinar a estimação de movimento. Esse refinamento poderia ocorrer de várias maneiras, a escolhida foi diminuir os possíveis tamanhos dos blocos dos macroblocos. Assim, em vez de blocos de 16×16 a 4×4 pixeis, os possíveis tamanhos de blocos serão de 16×16 a 1×1 (um pixel). A seguir, será criado um codificador que utiliza essa técnica para realizar a operação de casamento de padrões.

A operação de casamento de padrões realizada pelo codificador de vídeo CPST é efetuada pelo processo da estimação de movimento refinada, onde é possível processar blocos de 16×16 a 1×1 pixel. O "dicionário adaptativo" do processo de casamento de padrões continua o mesmo, formado por quadros já processados ante-

riormente. As palavras desse dicionário continuarão sendo os vetores de movimento, as listas e referências, porém com algumas modificações, como será mostrado na Seção 4.1. Assim, para cada bloco (16×16 a 1×1 pixel), é realizada uma busca pelo melhor casamento separadamente dos demais blocos, de acordo com o tipo de predição (lista 0, lista 1 ou bipredição).

No novo codificador CPST, para os *slices* dos tipos P e B, cada bloco de 16×16 a 1×1 pixel é codificado com vetores de movimento independentes. Entretanto, o codificador CPST não utilizará os vetores de movimento de blocos menores que 4×4 pixeis para o cálculo da predição vetorial. Pois, em blocos menores que 4×4 pixeis, por causa da pequena dimensão, o processo de estimação de movimento não está realizando a procura pelo "movimento", e sim pelo melhor casamento. Logo, em razão dessa sutil diferença, não faz sentido utilizar os vetores de movimento de blocos menores que 4×4 pixeis[1] na predição dos vetores de movimento dos blocos adjacentes. A predição de vetores de movimento para blocos menores que 4×4 pixeis ocorrerá da seguinte forma: será realizada uma predição de vetor de movimento para o bloco de tamanho 4×4 pixeis e todos os blocos menores contidos nesse mesmo bloco de 4×4 pixeis terão a mesma predição de vetor de movimento. Caso, entre os blocos codificados de um macrobloco, estiver um bloco de tamanho 4×4 pixeis, o vetor de movimento desse bloco poderá ser utilizado na predição de vetores de movimento de outros blocos adjacentes. Resumindo, na predição de vetores de movimento, serão utilizados apenas vetores de blocos maiores ou iguais a 4×4 pixeis.

Assim como não faz sentido utilizar os vetores de blocos menores que 4×4 na predição de vetores de movimento, não faz sentido usar o mesmo vetor de movimento da componente de luma (Y') para as componentes de croma (U e V). Deste modo, no codificador CPST, para os blocos de tamanhos 16×16, 8×16, 16×8, 8×8, 4×8 e 8×4 pixeis é codificado o mesmo vetor (ou vetores, no caso de bipredição) de movimento para as componentes de luma e de croma. Já para os blocos 4×4 a 1×1 pixel são codificados separadamente o vetor de movimento de cada componente: luma Y', croma U e croma V. Ao codificar blocos de tamanho 4×4, serão calculados, independentemente, vetores de movimento para cada componente de luma e de croma (U e V), mas será o vetor da componente de luma que será utilizado na predição de vetores de movimento dos blocos adjacentes.

Foi utilizada boa parte da norma H.264/AVC para a criação deste novo codificador, alterando apenas o necessário para a adição da estimação de movimento de blocos 4×4 a 1×1 pixel. As transformadas (DCT e a transformada de Hadamard) ainda serão utilizadas nos *slices* do tipo I. Apesar do fato de que o princípio do novo

[1]Por uma questão de simplificação, o processo de busca e a posição relativa dos blocos (original e predito) continuarão sendo chamados de estimação de movimento e vetor de movimento respectivamente, mesmo para blocos menores que 4×4 pixeis.

codificador seja criar um codificador de vídeo sem transformadas, a utilização das transformadas em *slices* do tipo I não anula esse princípio. Isto acontece porque, como se trata de um *slice* cuja codificação não depende dos demais *slices* já processados, ele poderia ser codificado de várias formas sem utilizar necessariamente as transformadas. Portanto, para simplificar, a codificação dos *slices* do tipo I no novo codificador não sofreu nenhuma modificação em relação aos *slices* desse tipo do padrão H.264/AVC, utilizando transformadas e somente blocos de tamanho 16×16 a 4×4 pixeis.

Vale ressaltar que a implementação do método CPST implica em modificações na norma H.264/AVC, e obviamente, a *bitstream* desse novo codificador não será compatível com os decodificadores H.264/AVC atuais.

4.1 Modificações Necessárias na Norma

Primeiramente, é necessário estabelecer como será o particionamento dos macroblocos, o qual seguirá o mesmo princípio do particionamento (estrutura de árvore) da norma H.264/AVC. O primeiro particionamento de macroblocos pode ser executado de 4 modos: 1 partição de 16×16 pixeis, 2 de 8×16 pixeis, 2 de 16×8 pixeis ou 4 de 8×8 pixeis. Caso escolhidas as 4 partições de 8×8 pixeis, cada partição de macrobloco de 8×8 pixeis pode ainda ser dividida independentemente em subpartições (chamadas de submacroblocos): 1 subpartição de 8×8 pixeis, 2 de 4×8 pixeis, 2 de 8×4 pixeis ou 4 de 4×4 pixeis. Caso escolhidas as 4 subpartições de 4×4 pixeis, então, seguindo a mesma estrutura, cada subpartição 4×4 pode ainda ser dividida independentemente em minipartições (chamadas de minimacroblocos): 1 minipartição de 4×4 pixeis, 2 de 2×4 pixeis, 2 de 4×2 pixeis ou 4 de 2×2 pixeis. Caso escolhidas as 4 minipartições de 2×2 pixeis, cada minipartição de 2×2 pixeis pode ainda ser dividida independentemente em micropartições (chamadas de micromacroblocos): 1 micropartição de 2×2 pixeis, 2 de 1×2 pixeis, 2 de 2×1 pixeis ou 4 de 1×1 pixel.

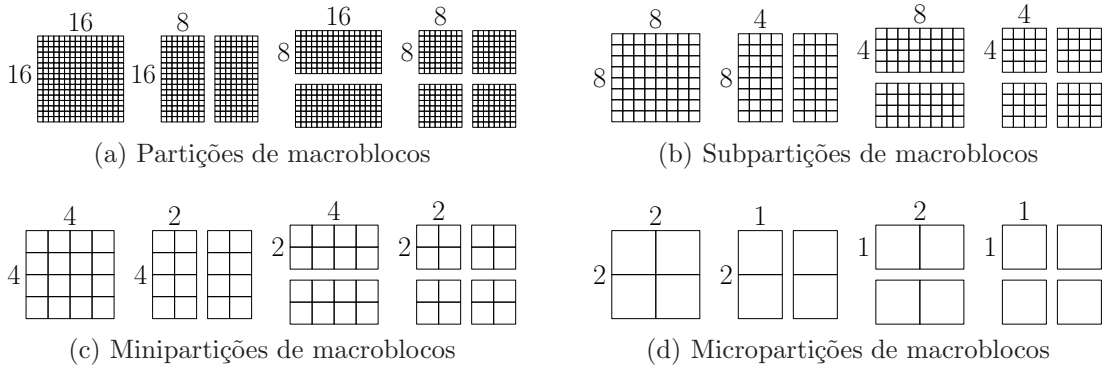
A Tabela 4.1 mostra os possíveis tamanhos de blocos de acordo com o tipo de particionamento de macrobloco, assim como a Figura 4.1 ilustra as possíveis divisões de blocos em um macrobloco, de 16×16 até 1×1 pixel, onde cada quadrado representa um pixel.

A seguir será explicado como é feita a codificação por entropia no codificador CPST. Devido à excelente taxa de compressão e relativa baixa complexidade do CABAC (em relação a codificadores aritméticos não-binários), o mesmo foi utilizado na codificação do tipo de particionamento, índice dos quadros de referência[2], listas

[2]O índice dos quadros de referência é o SE que informa qual a posição do quadro em uma determinada lista. Este é quadro o qual será utilizado como referência pelo processo estimação de

Tabela 4.1: Definições dos tamanhos de bloco para cada tipo de particionamento de macrobloco

Particionamentos de macrobloco	Tamanhos de bloco
Partições	16×16, 8×16, 16×8 e 8×8
Subpartições	8×8, 4×8, 8×4 e 4×4
Minipartições	4×4, 2×4, 4×2 e 2×2
Micropartições	2×2, 1×2, 2×1 e 1×1



(a) Partições de macroblocos

(b) Subpartições de macroblocos

(c) Minipartições de macroblocos

(d) Micropartições de macroblocos

Figura 4.1: Possíveis tipos de partições de macroblocos (4.1a), partições de submacroblocos (4.1b), partições de minimacroblocos (4.1c) e partições de micromacroblocos (4.1d).

(0 ou 1) e vetores de movimento no codificador CPST.

4.1.1 Codificação por Entropia – CABAC – no CPST

A binarização, contextos e codificação dos tipos de partições de macroblocos e submacroblocos permanecem iguais às da norma H.264/AVC [1]. Como os índices de referências são codificados apenas para partições de macroblocos (16×16 a 8×8 pixeis), esse SE é codificado da mesma forma estabelecida pela norma H.264/AVC [1]. O tipo de predição Inter[3] – lista 0, lista 1 ou bipredição – é codificado também apenas para partições de macroblocos, 16×16 a 8×8 pixeis. Assim, todos os submacroblocos, minimacroblocos e micromacroblocos contidos dentro de uma mesma partição de 8×8 pixeis terão o mesmo quadro de referência e o mesmo tipo de predição. Os vetores de movimento das partições de macroblocos e submacroblocos, permanecem com a mesma binarização (concatenação das binarizações unária truncada e binarização de terceira ordem de Exp-Golomb [1] [7]) e os mesmos contextos [1] em relação à norma H.264/AVC. Assim, a codificação por entropia está completa para macroblocos e submacroblocos. Então, para codificar os minimacroblocos e mi-

movimento.

[3]Somente para *slices* do tipo B, já que nos *slices* do tipo P somente existe, na predição Inter, predição com base nos quadros da lista 0.

Tabela 4.2: Tabela de binarização fixa das subpartições de macrobloco em *slices* do tipo P.

Tipo de *Slice*	Submacrobloco	*Bin string*		
P	8×8	1		
	8×4	0	0	
	4×8	0	1	1
	4×4	0	1	0
Índice do bin		0	1	2

cromacroblocos basta adicionar ao CABAC esses tipos de particionamentos e seus respectivos vetores de movimento. Primeiramente será explicada a binarização e, em seguida, os contextos utilizados pelo CABAC desses novos SEs.

Devido a estrutura de árvore, as tabelas de binarizações e contextos dos tipos de minimacroblocos e micromacroblocos podem ser criados de forma independente. A binarização escolhida para os tipos de partições de minimacroblocos e de micromacroblocos é a mesma binarização para os tipos de partições de macroblocos e de submacroblocos, ou seja, binarização por tabela fixa. Para cada tipo de particionamento – minimacroblocos e micromacroblocos – é necessário criar 4 palavras binárias (*bin string*), ou seja, uma tabela fixa indicando o valor de cada bin. Para simplificar a criação do codificador CPST, a binarização por tabela fixa dos tipos de particionamento de minimacroblocos e micromacroblocos foi criada com os mesmos valores da tabela de binarização dos submacroblocos dos *slices* do tipo P, mostrados na Tabela 4.2. Tanto em partições de macroblocos como em partições de submacroblocos, é esperado que os blocos menores sejam codificados mais frequentemente em altas taxas. Isto acontece porque a codificação de blocos menores implica em um aumento da taxa devido à codificação dos SEs apresentados (tipo de particionamento de macroblocos e submacroblocos). Por isso, é de se esperar que blocos menores sejam codificados mais frequentemente em altas taxas, onde o multiplicador de Lagrange λ tenha valor baixo. Em taxas altas, o custo de codificação obtido através do operador de Lagrange $D + \lambda R$, onde D é a distorção, tem uma menor influência da taxa R. Por esta razão é que aos blocos maiores de partições de macroblocos, submacroblocos, minimacroblocos e micromacroblocos são atribuídos as palavras binárias menores. As Tabelas 4.3 e 4.4 mostram os valores de cada bin para cada tipo de minipartições e micropartições respectivamente, para todos os *slices* (P e B) e todas as componentes (luma e croma).

Os contextos, como explicado anteriormente, são os SEs responsáveis por conter a probabilidade de codificação de cada bin que será utilizado pelo codificador aritmético binário. A atualização do contexto [1] [7] é feita assim que o codificador aritmético processa cada bin, dependendo apenas do valor do bin codificado e do

Tabela 4.3: Tabela de binarização fixa das minipartições de macrobloco em *slices* do tipo P e B.

Tipo de *Slice*	Minimacrobloco	*Bin string*		
P e B	4×4	1		
	4×2	0	0	
	2×4	0	1	1
	2×2	0	1	0
Índice do bin		0	1	2

Tabela 4.4: Tabela de binarização fixa das micropartições de macrobloco em *slices* do tipo P e B.

Tipo de *Slice*	Micromacrobloco	*Bin string*		
P e B	2×2	1		
	2×1	0	0	
	1×2	0	1	1
	1×1	0	1	0
Índice do bin		0	1	2

estado do contexto antes da codificação. O índice do bin, o tipo de *slice* e os SEs dos blocos vizinhos especificam qual contexto será utilizado para cada bin de cada SE do bloco a ser processado. Para as partições de macroblocos e submacroblocos, nos *slices* dos tipos P e B, os valores iniciais dos contextos permanecem os mesmos da norma H.264/AVC [1]. Os valores iniciais dos contextos de minimacroblocos e micromacroblocos, tanto em *slices* do tipo P como B e para todas as componentes (luma e croma), são iguais aos valores dos contextos dos submacroblocos em *slices* do tipo P [1]. Note que estas escolhas não foram ótimas, pois não houve nenhum levantamento prévio das estatísticas de minimacroblocos e micromacroblocos, entretanto esta foi uma forma de poupar tempo e simplificar a criação do codificador CPST. Assim, futuramente, a utilização de melhores estatísticas deverá ser investigada. Na norma H.264/AVC os valores iniciais dos contextos dos vetores de movimento independem do tipo de *slice* e do tipo de particionamento (partições de macroblocos e submacroblocos). Portanto, no codificador CPST, os contextos dos vetores de movimentos são inicializados da mesma forma que está descrita na norma H.264/AVC [1], porém separados por tipo de particionamento, tipo de *slice* e componente (luma e croma, no caso de minimacroblocos e micromacroblocos). Assim, para cada tipo de partição (de macroblocos a micromacroblocos), de *slice* e de componente (luma e croma, no caso de minimacroblocos e micromacroblocos) serão atribuídos um conjunto de contextos inicializados da mesma forma que na norma H.264/AVC para a codificação dos vetores de movimento.

4.2 Resultados Experimentais

A implementação desse novo codificador foi baseada no software de referência do padrão H.264/AVC, versão JM 16.2 (12/11/2009) [9]. Nessa nova versão foram corrigidos alguns *bugs* na estimação de movimento em relação a versão JM 15.0. Portanto para uma comparação mais justa, foi refeito, a partir do JM 16.2, o codificador DRSB, descrito na Seção 3.1, chamado de DRSB16.2.

As sequências simuladas nesta seção são as mesmas do Capítulo 3. Todas as simulações envolveram a codificação dos primeiros 99 quadros das sequências de teste *Akiyo* (resolução QCIF), *Foreman* (CIF), *Flower Garden* (SIF) e *Mobile & Calendar* (CIF), tendo as componentes de croma sido subamostradas segundo o padrão 4:2:0. Como as componentes de croma foram subamostradas no padrão 4:2:0, implica que um bloco de 4×4 de luma corresponderá a dois blocos de 2×2 de croma, correspondentes às componentes de croma U e V. Portanto, para as componentes de croma só existirão partições de micromacroblocos (2×2 a 1×1).

A configuração utilizada incluiu:

- Perfil *High*
- Codificação bipreditiva
- Transformadas 4×4 e 8×8 (utilizada apenas nos *slices* do tipo I para todos os codificadores e em *slices* do tipo P para os codificadores JM16.2 e DRSB16.2.
- 1 *slice* por quadro
- *Slices* Intra: somente o primeiro
- Lista 0: 5 quadros de referência (somente *slices* I ou P)
- Lista 1: 1 quadro de referência (somente *slices* P)
- GOP (*Group Of Pictures*): IBPBPBPBP...
- Modo Intra habilitado para blocos de *slices* Inter
- Otimização taxa-distorção com complexidade alta

O mesmo teste adicional foi realizado para a sequência *Foreman*, com a definição do GOP alterada para o padrão: IBBBPBBBP... Os valores do parâmetro de quantização (*Quantization Parameter* - QP) utilizados para controlar a taxa foram os mesmos dos Capítulo 3, como mostrados na Tabela 4.5.

A Figura 4.2 mostra a curva taxa-distorção para a componente de luma somente com dados dos *slices* do tipo P para a sequência de vídeo *Foreman*. Como era de

Tabela 4.5: Valores de QP utilizados.

Tipo de *Slice*	QP								
B	45	40	35	30	25	20	15	10	05
I e P	43	38	33	28	23	18	13	08	03

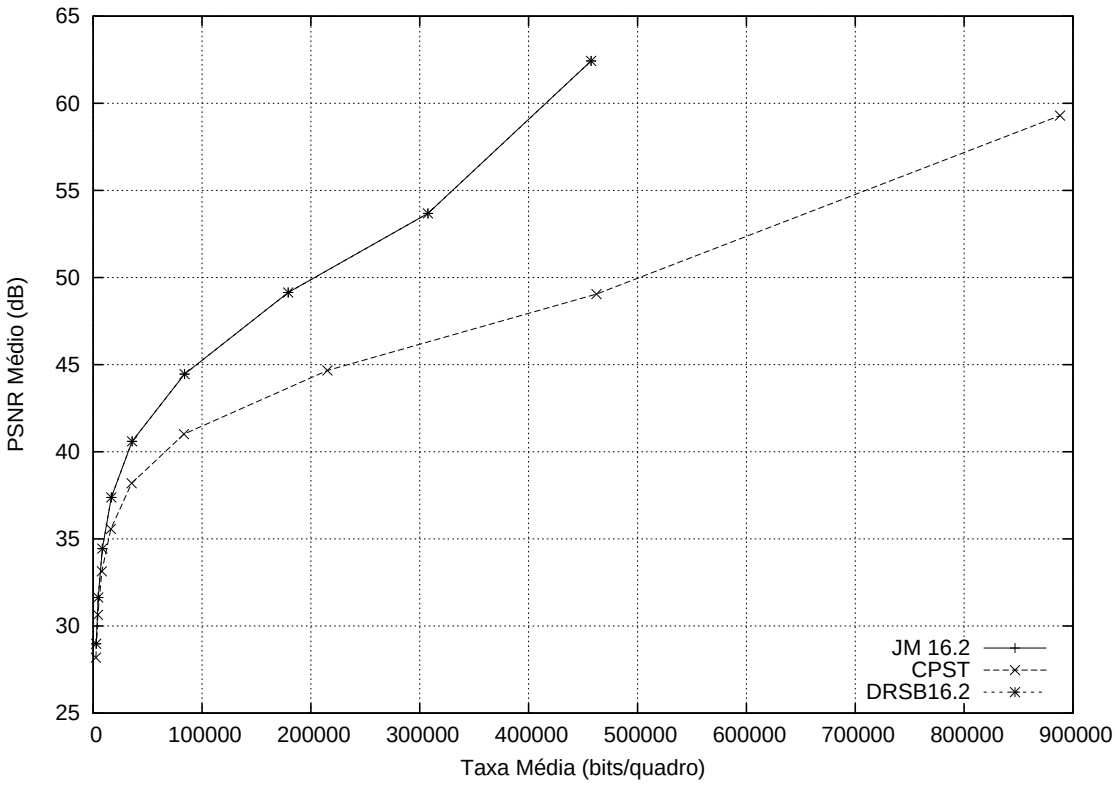
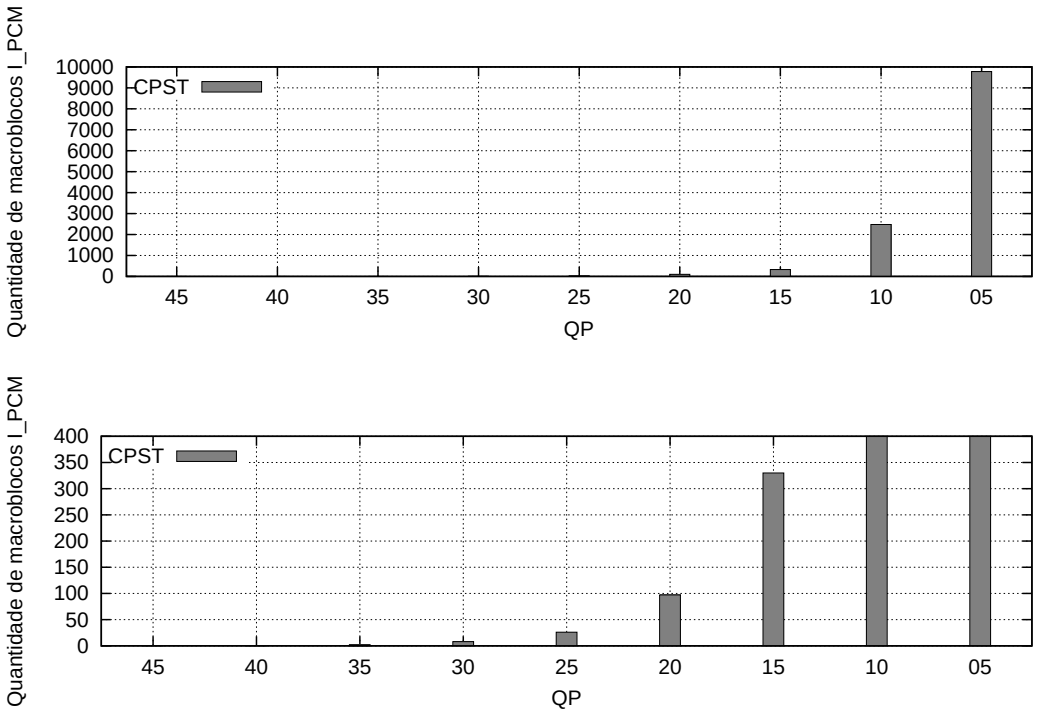
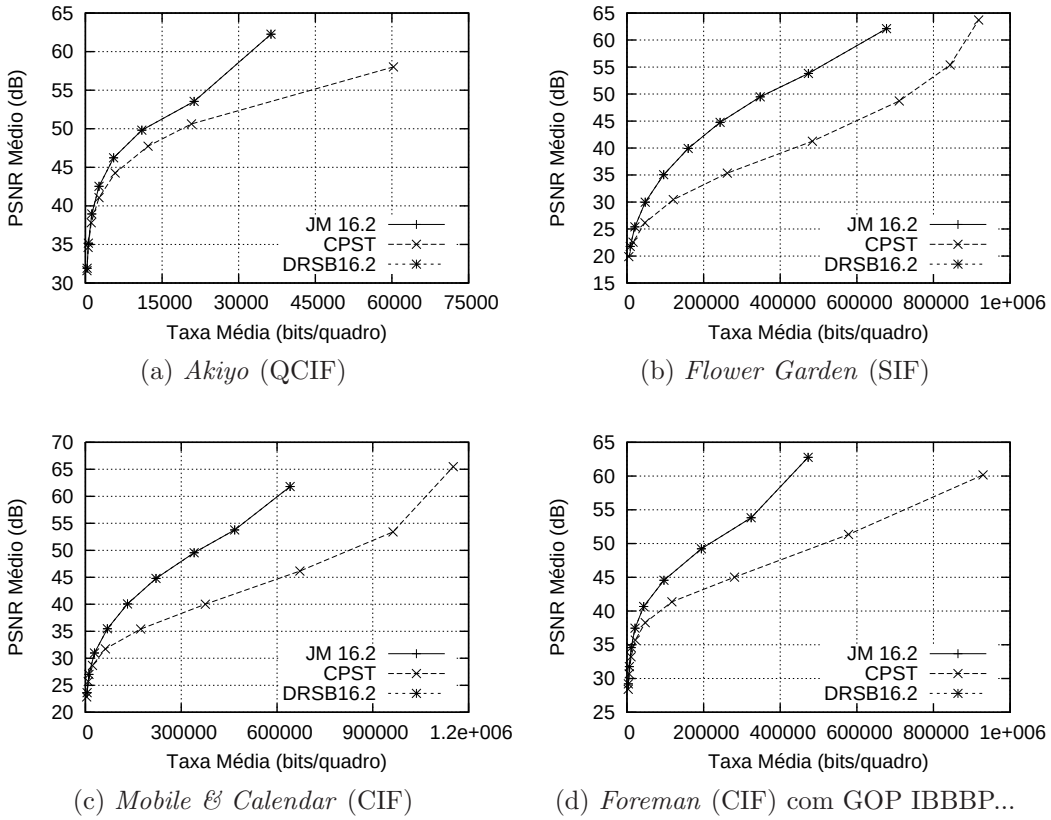


Figura 4.2: *Foreman* (CIF) - Somente quadros com *slices* do tipo P (Luma)



(b) Zoom

Figura 4.3: Quantidade de macroblocos I_PCM codificados na sequência *Foreman* (CIF) em *slices* do tipo P



(a) *Akiyo* (QCIF)

(b) *Flower Garden* (SIF)

(c) *Mobile & Calendar* (CIF)

(d) *Foreman* (CIF) com GOP IBBBP...

Figura 4.4: Somente quadros com *slices* do tipo P (Luma)

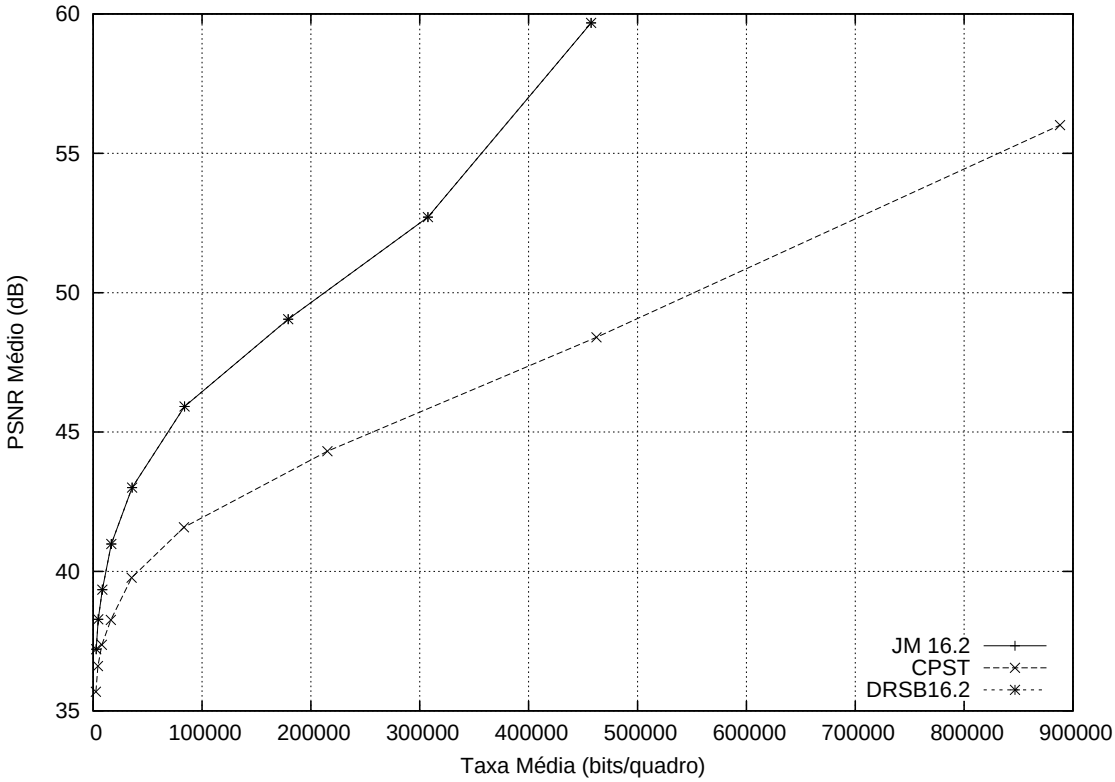


Figura 4.5: *Foreman* (CIF) - Somente quadros com *slices* do tipo P (Croma - U)

se esperar, os desempenhos dos codificadores JM 16.2 e DRSB16.2 são iguais, já que no codificador DRSB16.2 ainda são utilizados as transformadas para *slices* do tipo P. O codificador CPST possui um desempenho inferior ao do codificador JM 16.2 para *slices* do tipo P. Para *slices* desse tipo a estimação de movimento não tem muitas opções de predição Inter. Isto causa um aumento no número de blocos codificados com o modo I_PCM, aumentando, desse modo, significativamente a taxa. A quantidade de blocos I_PCM na codificação da sequência *Foreman* para *slices* do tipo P pode ser visualizada na Figura 4.3. Para os *slices* do tipo P, as curvas taxa-distorção das componentes de croma (U e V) possuem as mesmas características da componente de luma, como pode ser observado nas Figuras 4.5 e 4.6. Assim, para evitar o excesso de curvas, serão mostrados apenas as curvas de taxa-distorção das componentes de luma. As curvas das outras sequências de vídeo possuem o mesmo comportamento da sequência *Foreman*, e podem ser visualizadas no Apêndice A.3.

Para *slices* do tipo B, devido à baixa qualidade do "dicionário adaptativo" (*slices* do tipo P), o desempenho do codificador CPST é pior do que o codificador JM 16.2. É possível notar a falta de um bom dicionário ao verificar o desempenho do codificador DRSB16.2 em baixas e médias taxas, conforme mostrado na Figura 4.7. Porém, em altas taxas, mesmo com um dicionário de baixa qualidade o codificador CPST ultrapassa o desempenho do codificador DRSB16.2. Isto acontece devido à

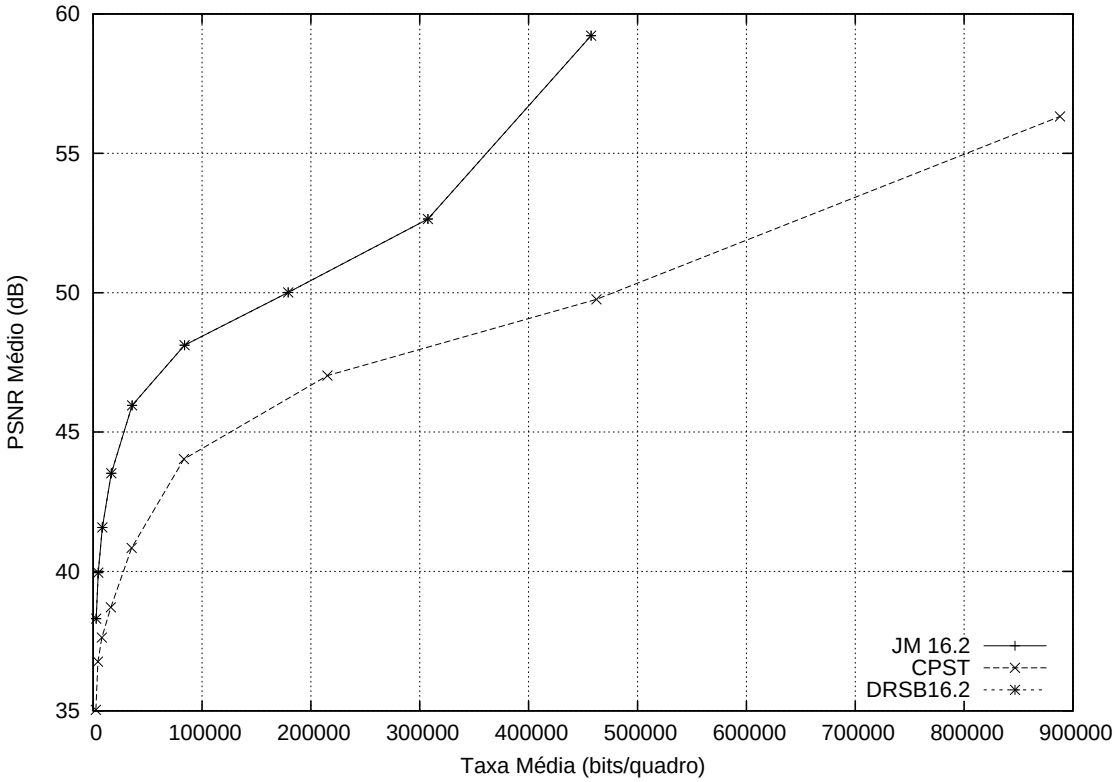


Figura 4.6: *Foreman* (CIF) - Somente quadros com *slices* do tipo P (Croma - V)

estimação de movimento refinada com blocos 4×4 a 1×1 pixel, que passam a ser codificados mais frequentemente nessas taxas. A estimação de movimento refinada diminui a quantidade de macroblocos com codificação I_PCM, como pode ser observado na Figura 4.8. Por exemplo, a quantidade de macroblocos com codificação I_PCM no codificador DRSB16.2 para um QP igual a 05 foi de 9168, ao passo que, no codificador CPST, apenas 362 macroblocos I_PCMs foram utilizados. Esta redução é bastante significativa, efeito que tem consequência direta no desempenho do codificador CPST em altas taxas.

Ao verificar as curvas taxa-distorção do todos os quadros juntos é possível notar a grande perda de desempenho do codificador CPST causada pela perda de desempenho do mesmo nos *slices* do tipo P. A estimação de movimento refinada não é suficiente para criar um codificador de casamento de padrões sem transformadas com o mesmo desempenho do codificador H.264/AVC. Serão necessárias algumas modificações para que o codificador CPST possa ser aprimorado para não ter uma perda de desempenho tão grande. Uma possível modificação poderia ser a criação de um dicionário adaptativo mais refinado. Por exemplo, adicionando quadros criados a partir dos quadros armazenados de acordo com algum algoritmo enriquecendo o dicionário e, consequentemente, a busca feita pelo processo de estimação de movimento. Diminuindo, dessa forma, a quantidade de macroblocos com codificação I_PCM nos *slices* do tipo P. Outra possível solução, seria a criação de um algoritmo

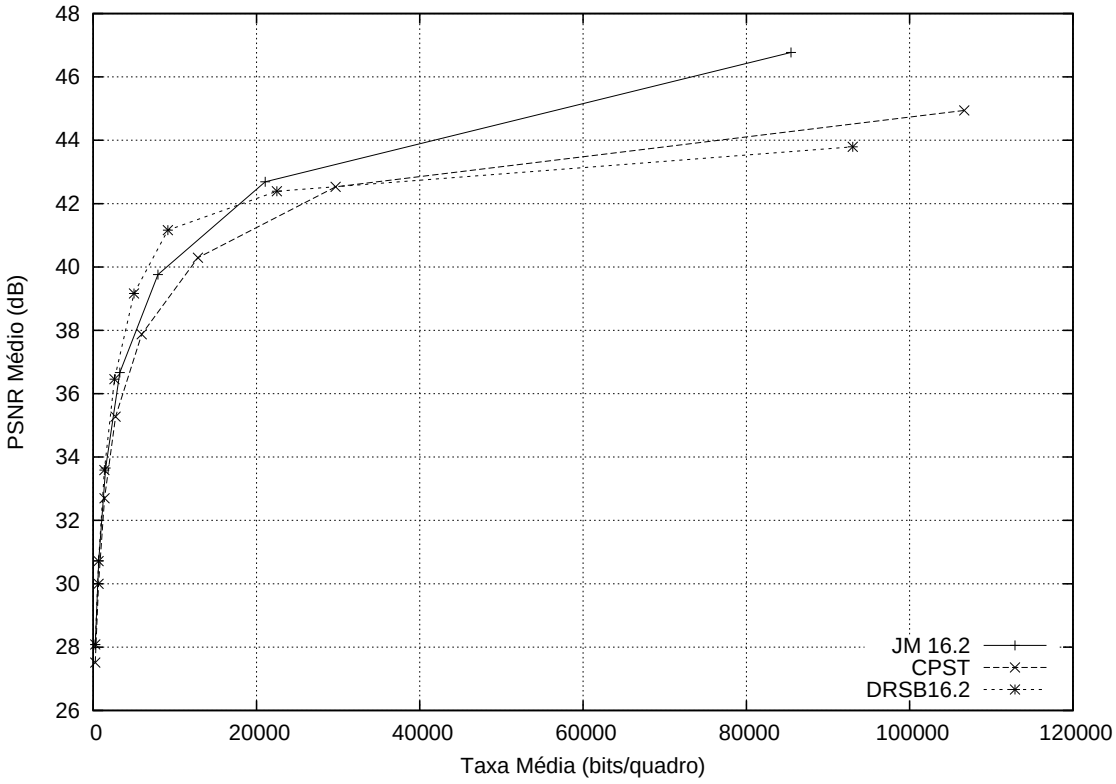
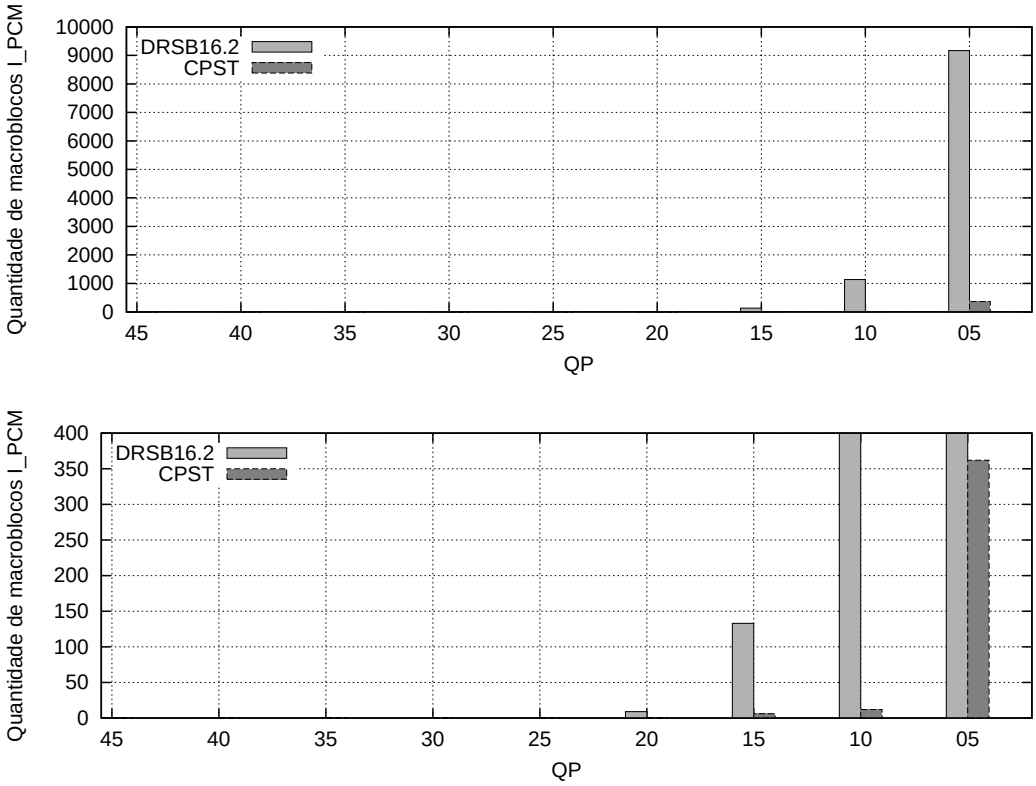


Figura 4.7: *Foreman* (CIF) - Somente quadros com *slices* do tipo B (Luma)



(b) Zoom

Figura 4.8: Quantidade de macroblocos I_PCM codificados na sequência *Foreman* (CIF) em *slices* do tipo B

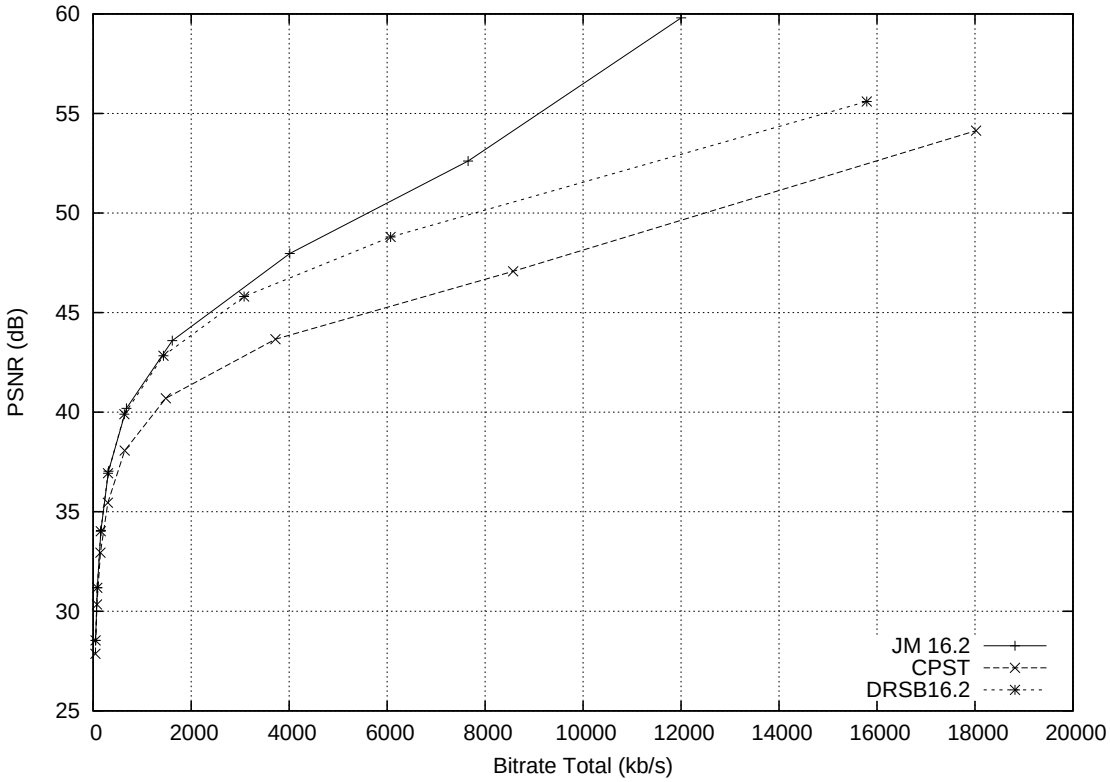


Figura 4.9: *Foreman* (CIF) - Todos os quadros (Luma)

que obtivesse distorção nula em blocos de tamanho 1×1 pixel, ao invés de utilizar estimação/compensação de movimento nesses blocos.

Para as outras sequências de vídeo, *Akiyo* (QCIF), *Flower Garden* (SIF) e *Mobile & Calendar* (CIF), os codificadores se comportaram de maneira bem semelhante ao observado para a sequência de vídeo *Foreman*. Isto pode ser observado nas Figuras 4.10, 4.11 e 4.12. Uma exceção é dada talvez pela sequência *Akiyo*, onde o codifcador CPST obteve o melhor desempenho entre os 3 para altas taxas. Porém, por tratar-se de uma sequência com pouco movimento, a mesma favorece os codificadores baseados apenas em casamento de padrões.

Ao mudar o GOP da sequência de vídeo *Foreman*, de IBP para IBBBP, os três codificadores permaneceram com o mesmo comportamento, o que indica uma possível independência de desempenho em relação ao tipo de GOP. Os resultados para essa sequência nesses GOPs podem ser observados nas Figuras 4.7 e 4.13.

Neste capítulo, foi mostrada uma primeira abordagem para a criação de um codificador completamente independente de transformadas, mudando o paradigma de transformadas seguida de quantização para codificação de vídeo. Infelizmente não é uma tarefa simples superar o desempenho do H.264/AVC, mas os resultados experimentais do codificador DRSB16.2 juntamente com os do codificador CPST indicam um possível caminho para esse objetivo ao adicionar um dicionário adaptativo mais refinado. No próximo capítulo será apresentada a conclusão da dissertação.

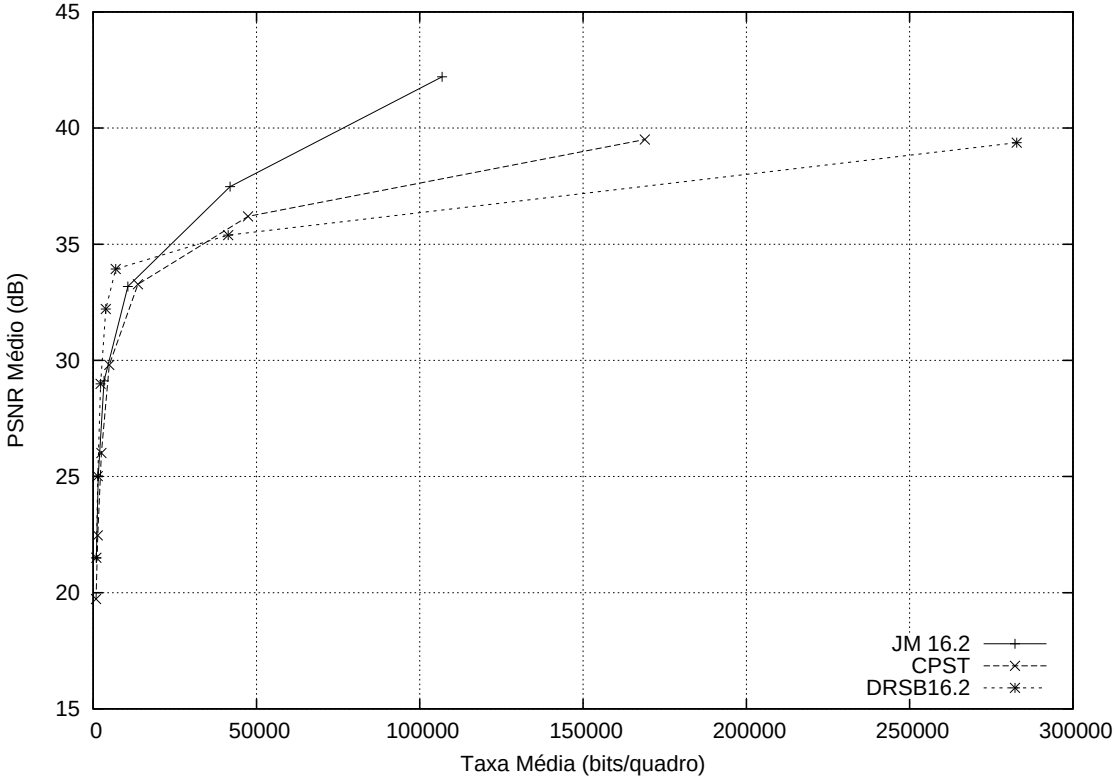


Figura 4.10: *Flower Garden* (SIF) - Somente quadros com *slices* do tipo B (Luma)

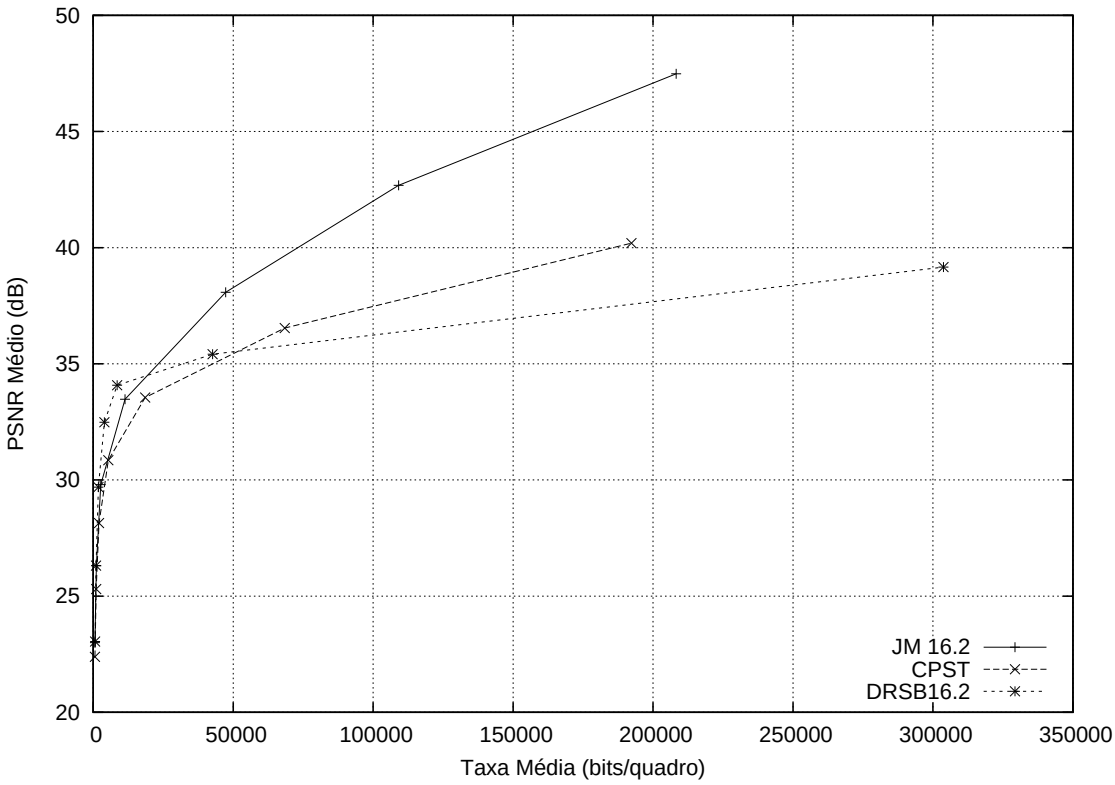


Figura 4.11: *Mobile & Calendar* (CIF) - Somente quadros com *slices* do tipo B (Luma)

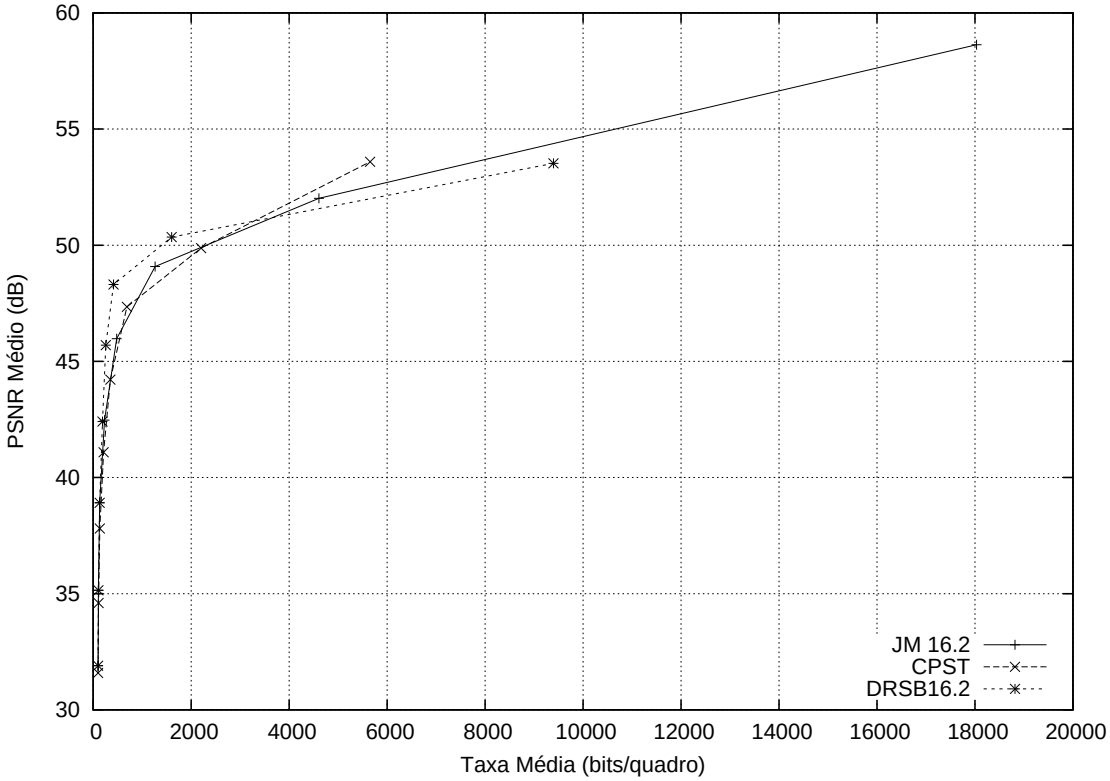


Figura 4.12: *Akiyo* (QCIF) - Somente quadros com *slices* do tipo B (Luma)

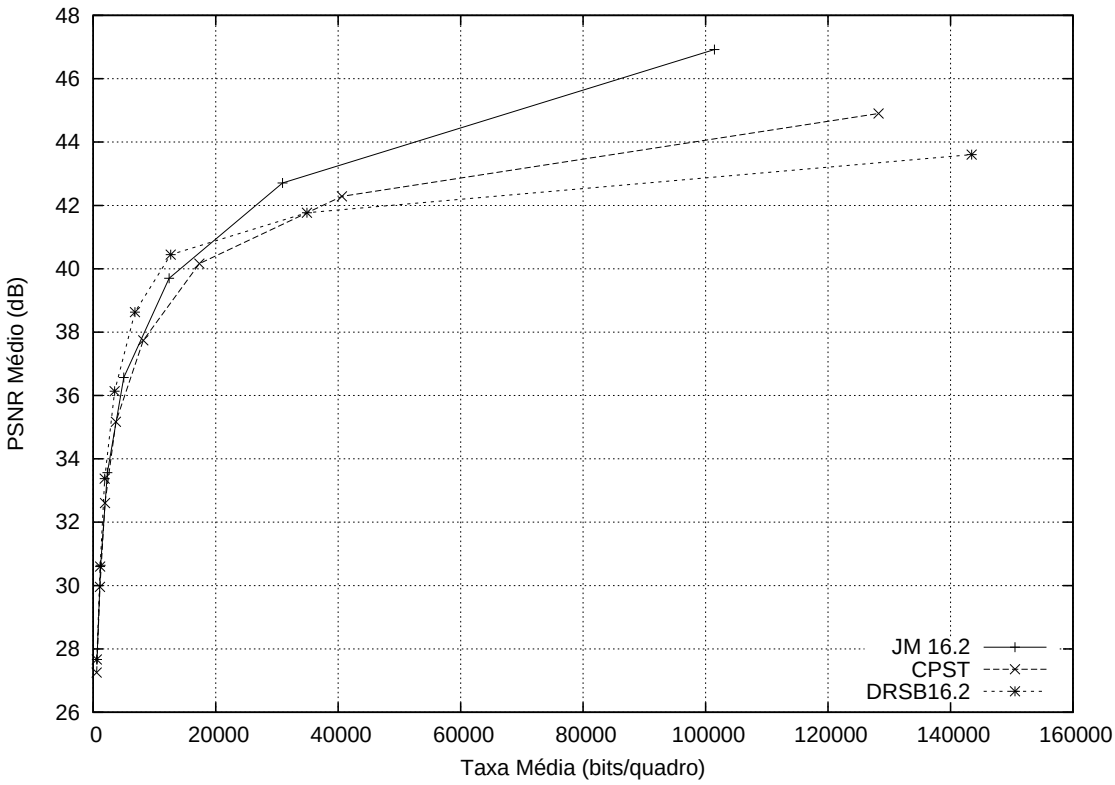


Figura 4.13: *Foreman* (CIF) - Somente quadros com *slices* do tipo B com GOP IBBBP (Luma)

Capítulo 5

Conclusões

Neste capítulo serão apresentadas todas as as contribuições deste trabalho, fazendo uma breve revisão de cada capítulo. Na Seção 5.1, serão apresentadas as sugestões para futuros trabalhos para a codificação de vídeo utilizando casamento de padrões.

O codificador H.264/AVC, como dito anteriormente, é o estado da arte em codificação de vídeo. Foram anos de pesquisa, desde 1990[1], para atingir seu alto grau de desempenho, já que o padrão H.264/AVC é um melhoramento dos codificadores antigos (H.261, MPEG-1 Part 2, H.262/MPEG-2 Part 2, H.263 e MPEG-4 Part 2). Levando este fato em consideração, não é uma tarefa simples criar um codificador de vídeo que supere o desempenho do codificador H.264/AVC.

O objetivo inicial desta dissertação era construir um codificador de vídeo livre de transformadas e com base no casamento de padrões. Assim, o objetivo foi dividido em duas partes. Na primeira parte, Capítulo 3, o objetivo era criar um codificador com base no casamento de padrões porém ainda compatível com a norma H.264/AVC. Na segunda parte, Capítulo 4, o objetivo era criar um novo codificador de vídeo livre de transformadas, porém utilizando diversas ferramentas da norma H.264/AVC, entre elas o CABAC.

No Capítulo 3, ao implementar o casamento de padrões através da estimação de movimento, ou seja, ao descartar os resíduos da predição em *slices* do tipo B obteve-se um melhor desempenho em relação ao codificador H.264/AVC para determinadas taxas. Ao criar o codificador com Descarte de Resíduos em *Slices* do tipo B (DRSB) obteve-se um melhor desempenho, em baixas taxas, em relação ao codificador H.264/AVC, como pode ser observado na Seção 3.1. Como trata-se de uma modificação que sempre elimina os resíduos em *slices* do tipo B, houve uma diminuição na complexidade computacional desses *slices* do codificador DRSB em relação ao codificador H.264/AVC, devido ao não uso de transformadas. Por causa dos ganhos obtidos com codificador DRSB em baixas taxas, foi criado um novo co-

[1] Ano de lançamento do padrão H.261.

dificador compatível com a norma e que obteve um desempenho sempre melhor ou igual ao codificador H.264/AVC em todas as taxas. A meta desse novo codificador é alcançar o desempenho do codificador DRSB para baixa taxa e o do codificador H.264/AVC para alta taxa. Assim, na Seção 3.2 é descrita a criação desse novo codificador, o codificador com Otimização na Codificação de *Slices* do tipo B (OCSB). No codificador OCSB, para cada bloco dos *slices* do tipo B é calculado o custo com e sem resíduo, e aquele que alcançar o menor custo será o escolhido. Como trata-se de uma modificação que apenas acrescenta um novo modo de codificação de bloco (eliminando os resíduos), a mesma pode ser paralelizada, aumentando deste modo a complexidade computacional, porém não prejudicando a velocidade de codificação. Portanto, obteve-se um codificador com um desempenho melhor ou igual ao H.264/AVC, dependendo da taxa, e com uma complexidade computacional um pouco maior, porém compatível com o *hardware* atual.

No capítulo 4, foi criado um codificador que utiliza apenas o casamento de padrões, eliminando completamente as transformadas dos *slices* dos tipos B e P. A forma que foi escolhida para implementar o casamento de padrões foi utilizar uma estimação de movimento mais refinada, inserindo blocos menores que podem ir até 1×1 pixel. Assim, foi criado o codificador com Casamento de Padrões Sem Transformadas (CPST). Contudo, a estimação de movimento em *slices* do tipo P é limitada, pois utiliza apenas a predição com base nos quadros da lista 0. Este fato reduziu consideravelmente a qualidade nos quadros que utilizam esses *slices*. Outro fator que contribuiu para a perda de desempenho dos *slices* do tipo B e P no CPST foi seu baixo desempenho para os blocos menores. Pois, ao utilizar os blocos menores, é esperado que a taxa aumente devido ao *overhead* causado pela codificação do particionamento, porém a distorção nesses blocos não diminuiu o necessário. Em blocos menores que 4×4 pixeis, o aumento da taxa devido a codificação do particionamento leva a uma limitação nos tamanhos de vetores de movimento, visto que vetores com valores altos provocam um maior aumento na taxa e, consequentemente, no custo. Assim, quanto menor o bloco, maior será sua taxa para a codificação de particionamento e menor será a faixa dinâmica permitida para dos vetores de movimentos. Para um bom codificador, é esperado que à medida que a taxa suba a distorção diminua. No codificador H.264/AVC, ao codificar em altas taxas, a distorção diminui com a diminuição do SE QP que controla o passo de quantização dos coeficientes das transformadas. No codificador CPST, a diminuição da distorção depende da utilização de blocos cada vez menores, porém com o limite imposto na faixa dinâmica dos vetores de movimento dos blocos menores nem sempre é possível diminuir a distorção. Ao codificar blocos de tamanho 1×1 pixel, era esperado obter a distorção nula, mas isto não aconteceu devido ao limite dos vetores de movimento por causa do custo de codificação.

Vale citar que durante este trabalho identificamos 5 *bugs* no software de referência JM 16.2, especialmente na parte de estimação de movimento. Através do *maillist JVT Experts*, 3 *bugs* foram confirmados e corrigidos na nova versão JM 17.0 pelos programadores do software.

5.1 Trabalhos Futuros

Nesta seção serão discutidos alguns projetos futuros para codificação de vídeo sem a utilização da transformada.

O primeiro passo é obter a distorção nula em blocos de tamanho 1×1 pixel, obtendo, desta maneira, um melhor desempenho em altas taxas. Uma forma eficiente de conseguir este objetivo já está sendo estudada, podendo utilizar até mesmo a codificação I_PCM, ou seja, enviar o próprio valor do pixel.

Outro ponto importante é melhorar a codificação em *slices* do tipo P, devido ao baixo desempenho obtido nesses *slices* do codificador CPST. A forma mais viável talvez seja um melhoramento no dicionário adaptativo. Por exemplo, pode-se adicionar quadros diferentes dos já processados de acordo com uma certa regra ou utilizar um dicionário para codificação de vídeo inspirado no algoritmo MMP [4].

Por fim, poderá ser estudada uma forma de adicionar os bons desempenhos conseguidos, como por exemplo do codificador OCSB, na codificação de vídeo 3D.

Referências Bibliográficas

[1] *Draft of Version 4 of H.264/AVC (ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 part 10) Advanced Video Coding*. Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, January 2005. JVT-N050d1.

[2] D. MARPE, T. WIEGAND, AND S. GORDON. "H.264/MPEG4-AVC fidelity range extensions: tools, profiles, performance, and application areas". In: *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, Genoa, Italy, September 2005.

[3] A. GERSHO AND R.M. GRAY. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[4] M.B. DE CARVALHO. *Compressão de sinais multi-dimensionais usando recorrência de padrões multiescalas*. Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2001.

[5] T. WIEGAND, G.J. SULLIVAN, G. BJØNTEGAARD, AND A. LUTHRA. "Overview of the H.264/AVC video coding standard", *Circuits and Systems for Video Technology, IEEE Transactions on*, v. 13, n. 7, pp. 560–576, July 2003.

[6] I.E.G. RICHARDSON. *H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia*. Wiley, 2003.

[7] D. MARPE, H. SCHWARZ, AND T. WIEGAND. "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard", *Circuits and Systems for Video Technology, IEEE Transactions on*, v. 13, n. 7, pp. 620–636, July 2003.

[8] FLIERL, M., GIROD, B. "Generalized B pictures and the draft H.264/AVC video-compression standard", *Circuits and Systems for Video Technology, IEEE Transactions on*, v. 13, n. 7, pp. 587–597, July 2003.

[9] JOINT VIDEO TEAM (JVT) OF ISO/IEC MPEG AND ITU-T VCEG. "H.264/Advanced Video Coding Reference Software". Disponível em: <http://iphome.hhi.de/suehring/tml/download/old_jm/>.

[10] N.M.M. RODRIGUES, E.A.B. DA SILVA, M.B. DE CARVALHO, S.M.M. DE FARIA, AND V.M.M. DA SILVA. "On overriding H.264/AVC B-slice predicted residue coding". In: *Picture Coding Symposium (PCS)*, Lisbon, Portugal, November 2007.

[11] D.F. DE SOUZA, E.A.B. DA SILVA, A.J.S. DUTRA, N.M.M. RODRIGUES, S.M.M. DE FARIA, AND V.M.M. DA SILVA. "Otimização taxa-distorção para quadros-B no padrão H.264/AVC". In: *XXVII Simpósio Brasileiro de Telecomunicações (SBrT)*, Blumenau - SC, Brazil, September 2009.

Apêndice A

Figuras Complementares

A.1 DRSB

A.1.1 Todos os Quadros

Gráficos contendo informações de todos os quadros de uma mesma sequência para o método DRSB.

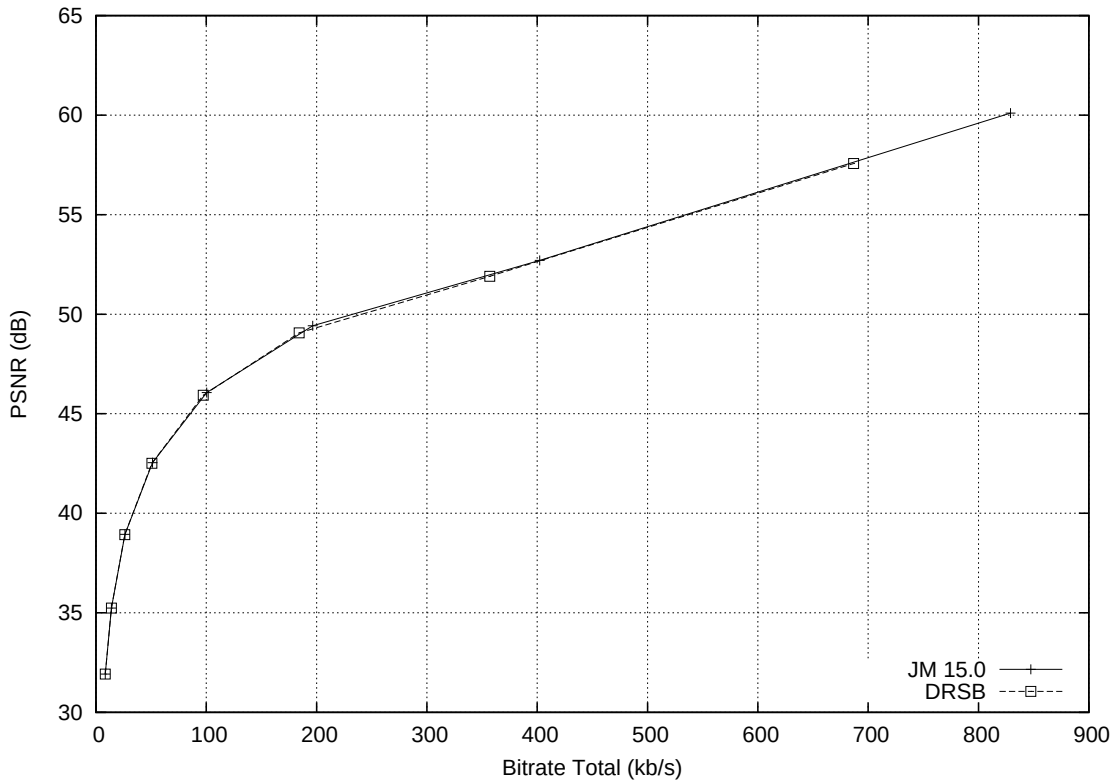


Figura A.1: *Akiyo* (QCIF) - Todos os quadros (Luma)

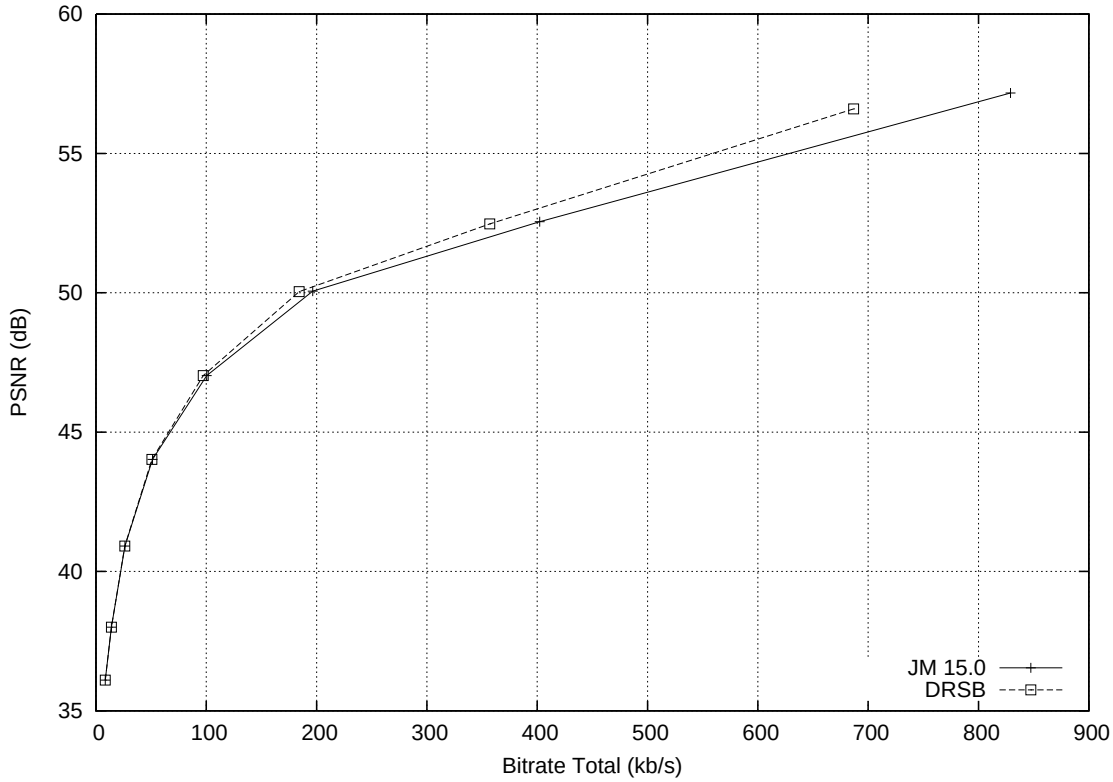


Figura A.2: *Akiyo* (QCIF) - Todos os quadros (Croma - U)

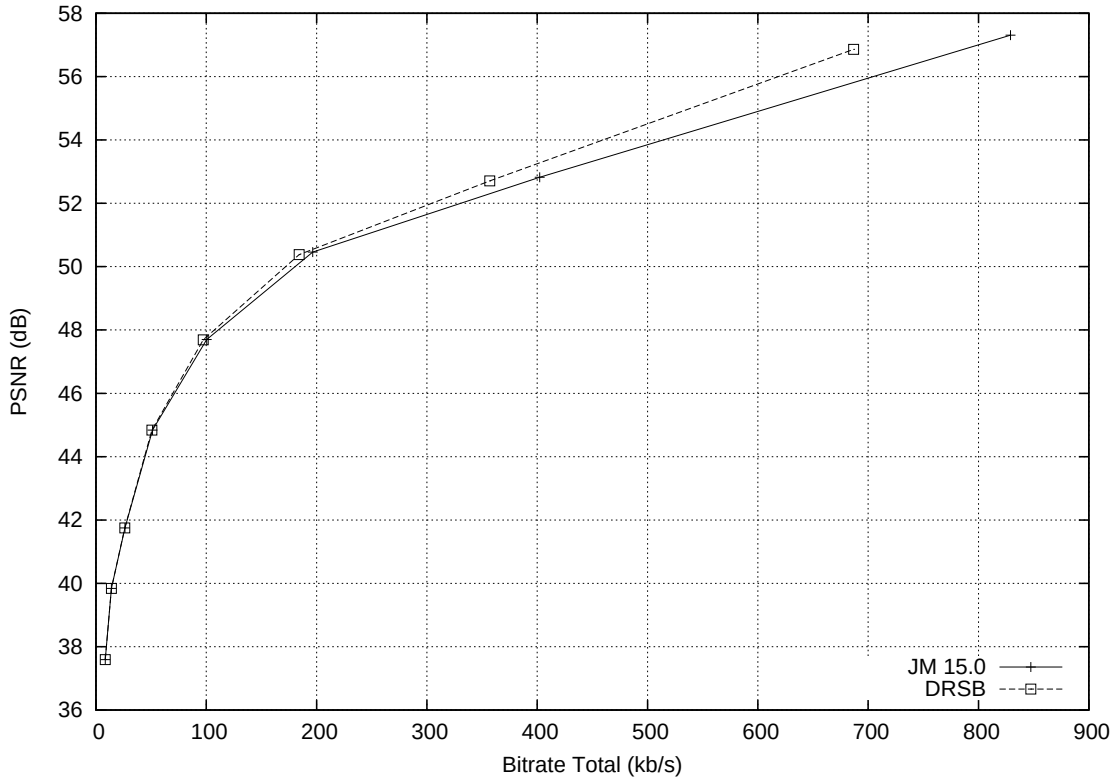


Figura A.3: *Akiyo* (QCIF) - Todos os quadros (Croma - V)

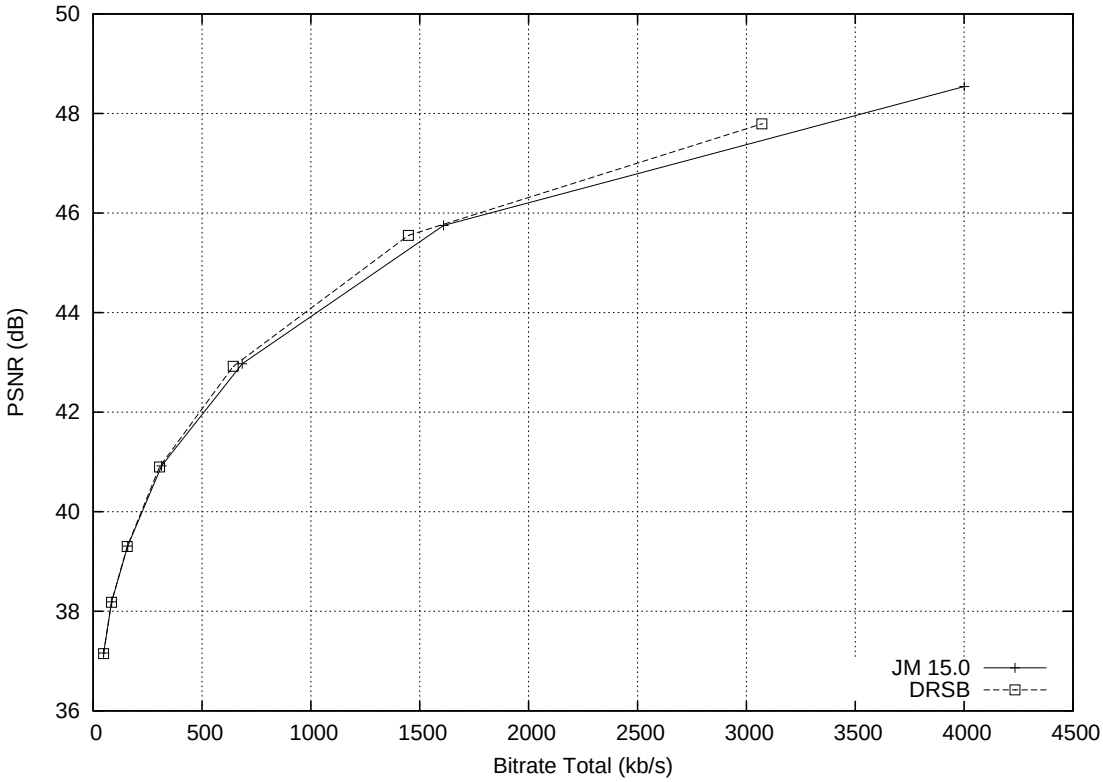


Figura A.4: *Foreman* (CIF) - Todos os quadros (Croma - U)

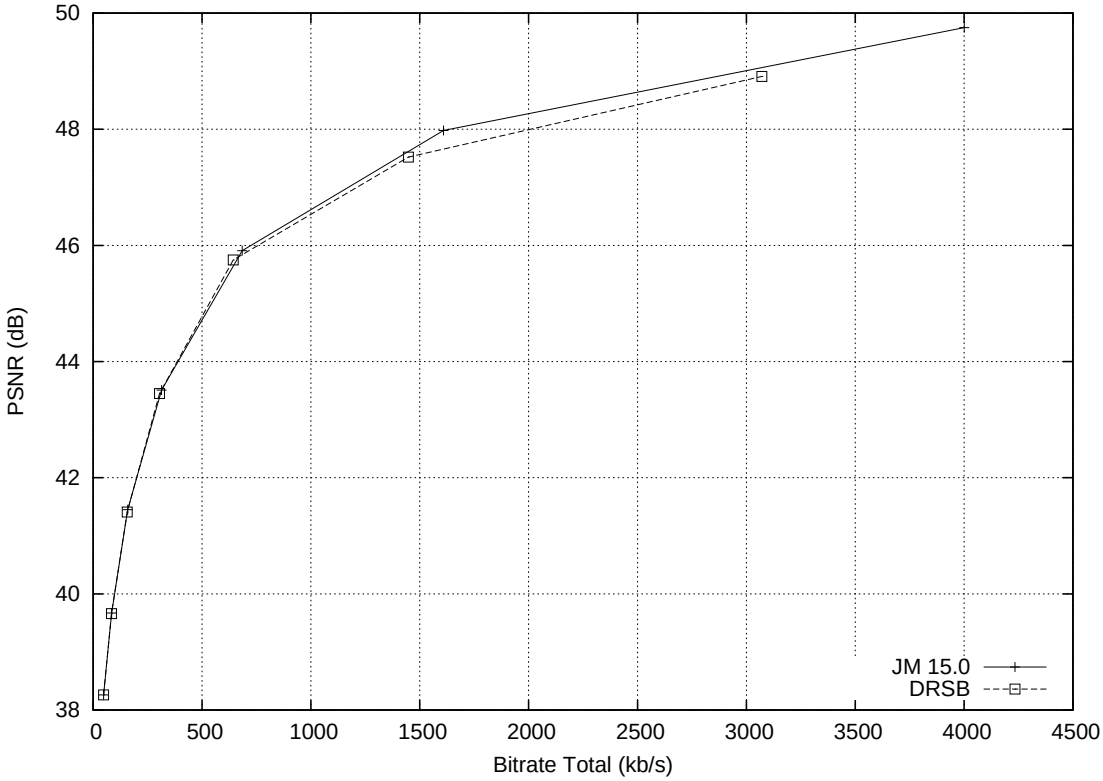


Figura A.5: *Foreman* (CIF) - Todos os quadros (Croma - V)

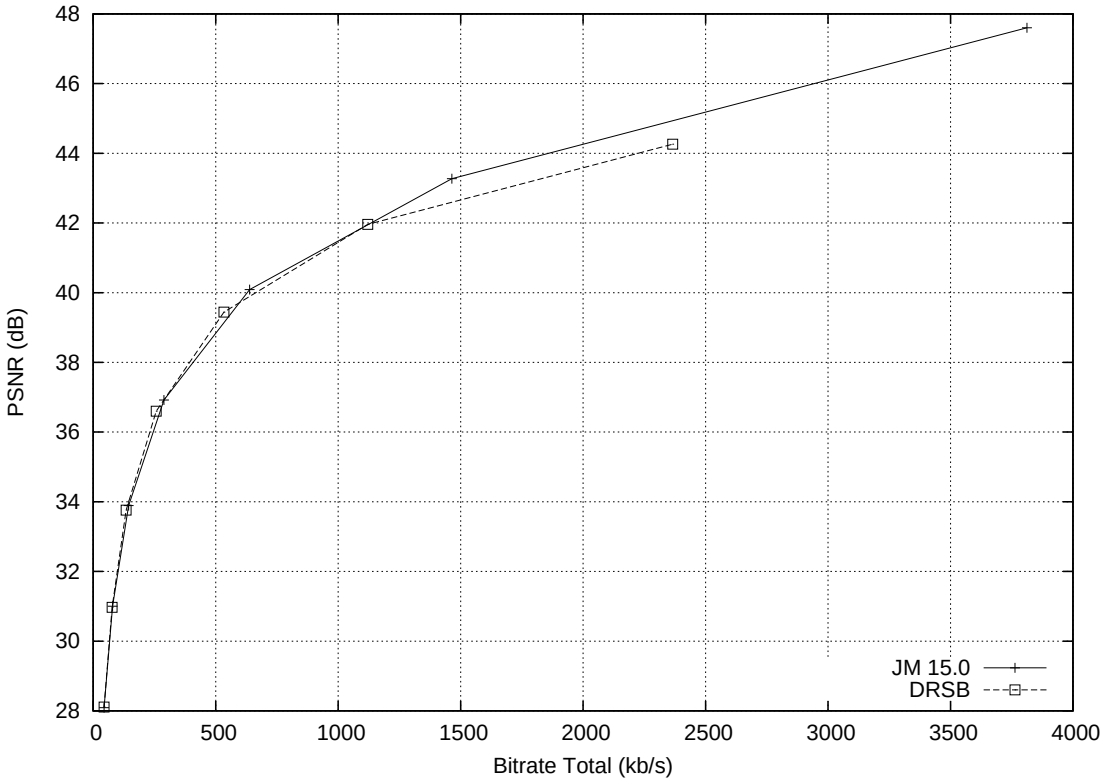


Figura A.6: *Foreman* (CIF) - Todos os quadros com GOP IBBBP (Luma)

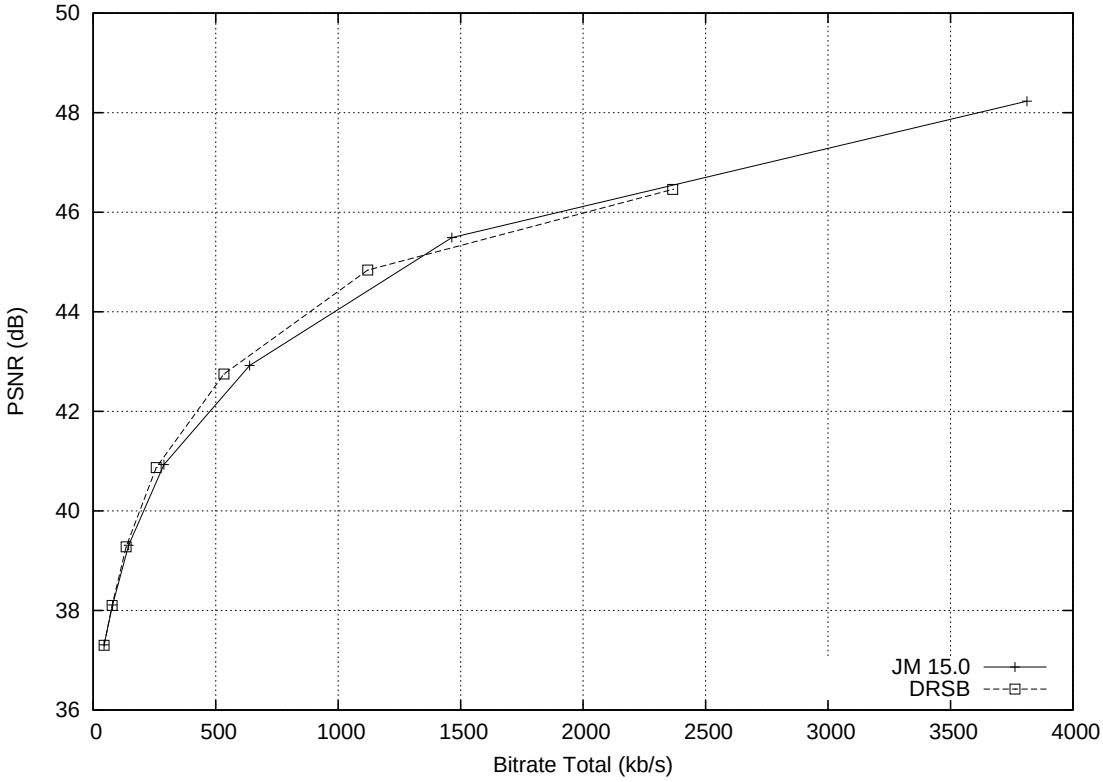


Figura A.7: *Foreman* (CIF) - Todos os quadros com GOP IBBBP (Croma - U)

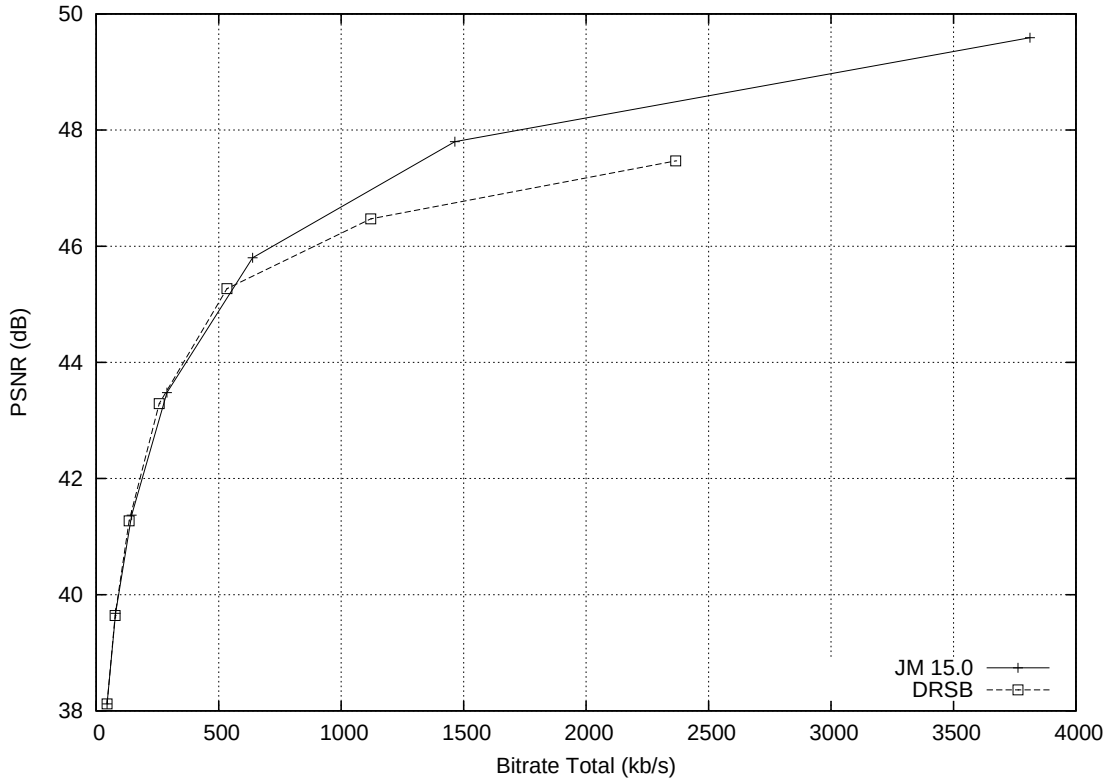


Figura A.8: *Foreman* (CIF) - Todos os quadros com GOP IBBBP (Croma - V)

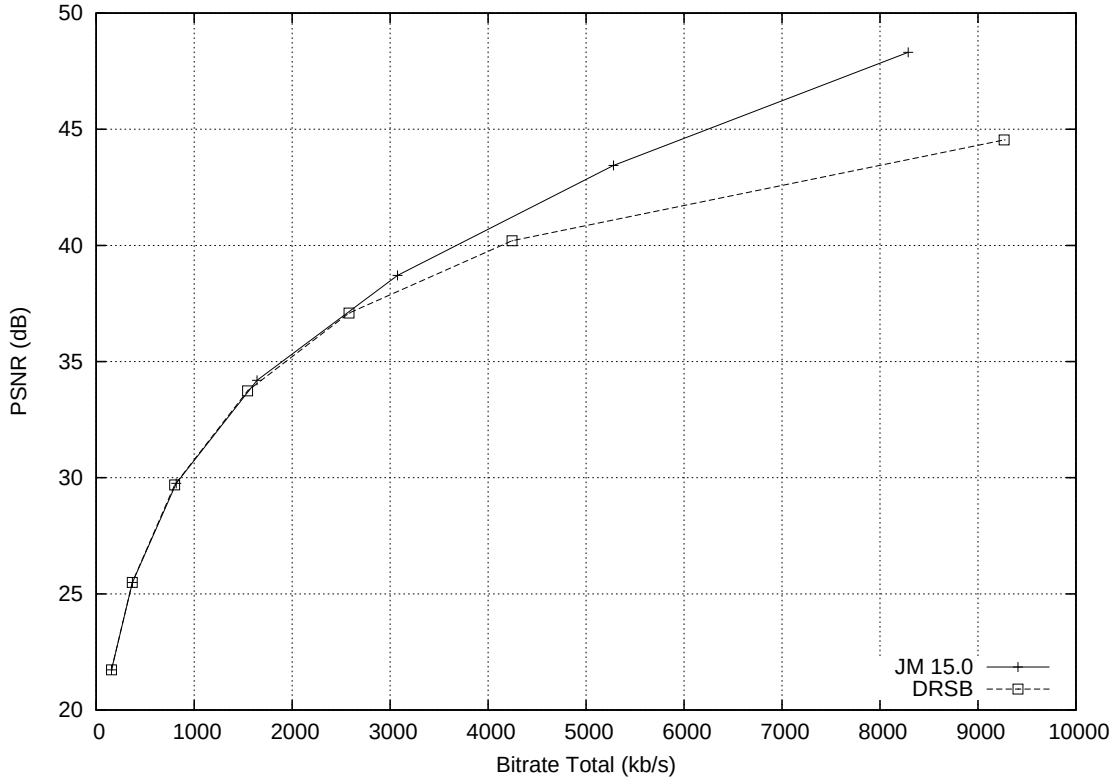


Figura A.9: *Flower Garden* (SIF) - Todos os quadros (Luma)

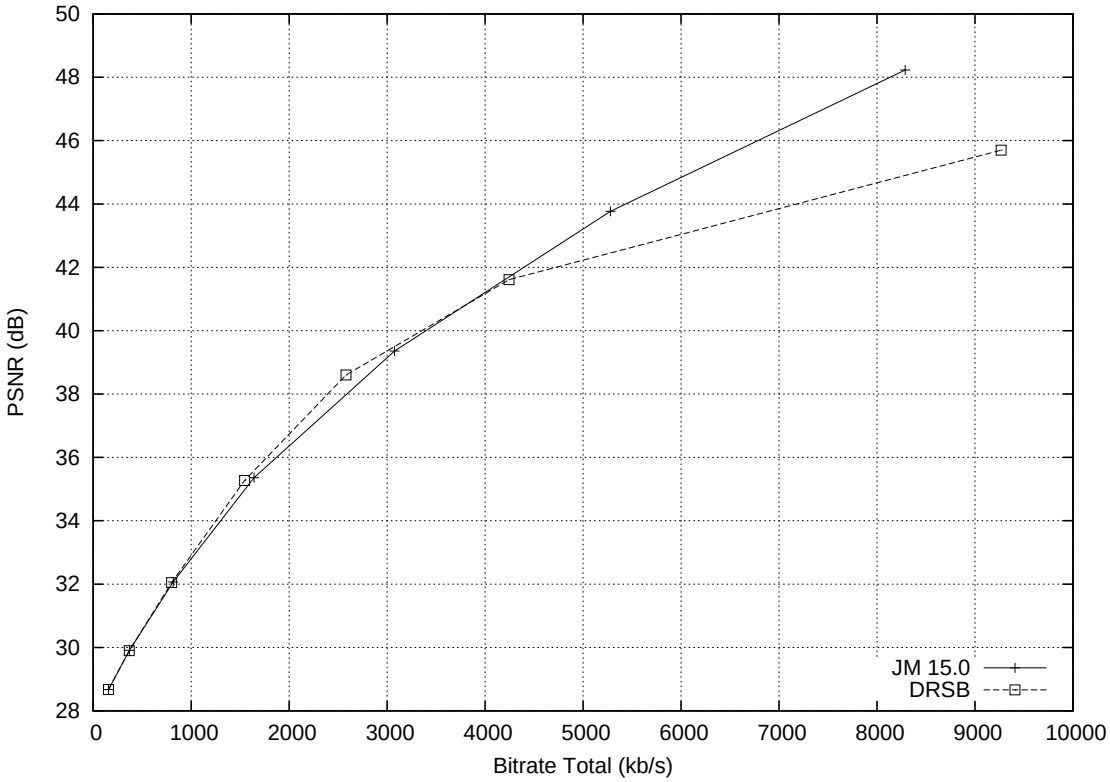


Figura A.10: *Flower Garden* (SIF) - Todos os quadros (Croma - U)

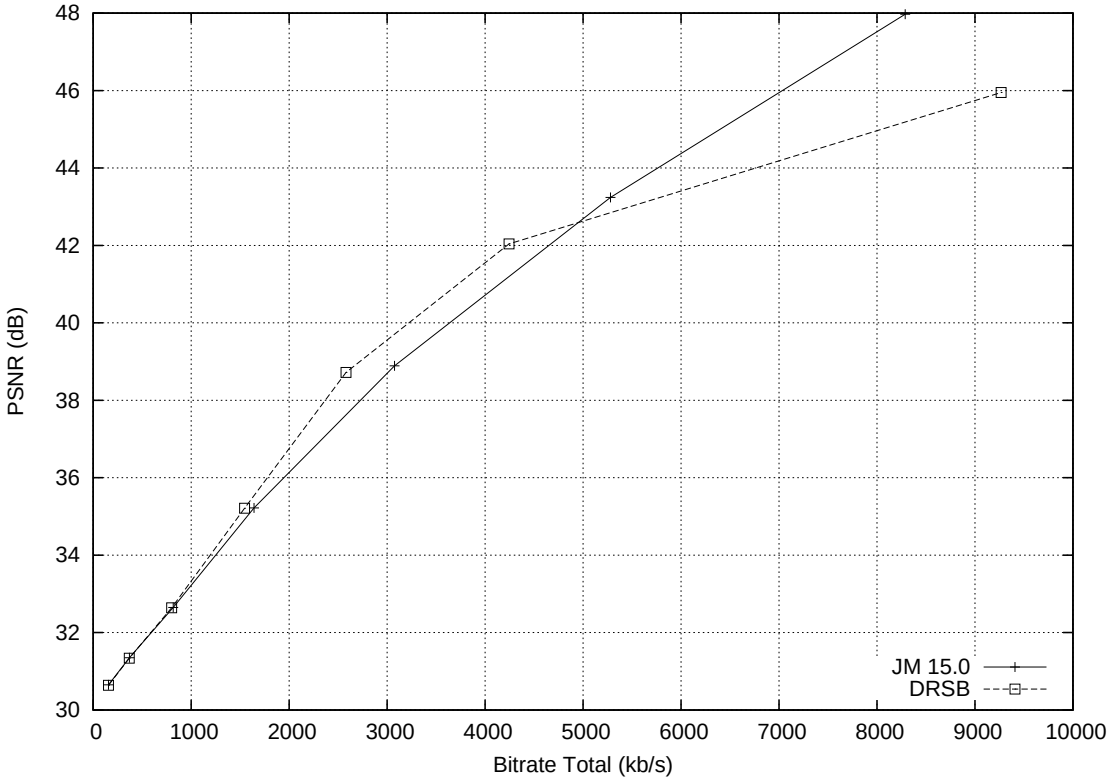


Figura A.11: *Flower Garden* (SIF) - Todos os quadros (Croma - V)

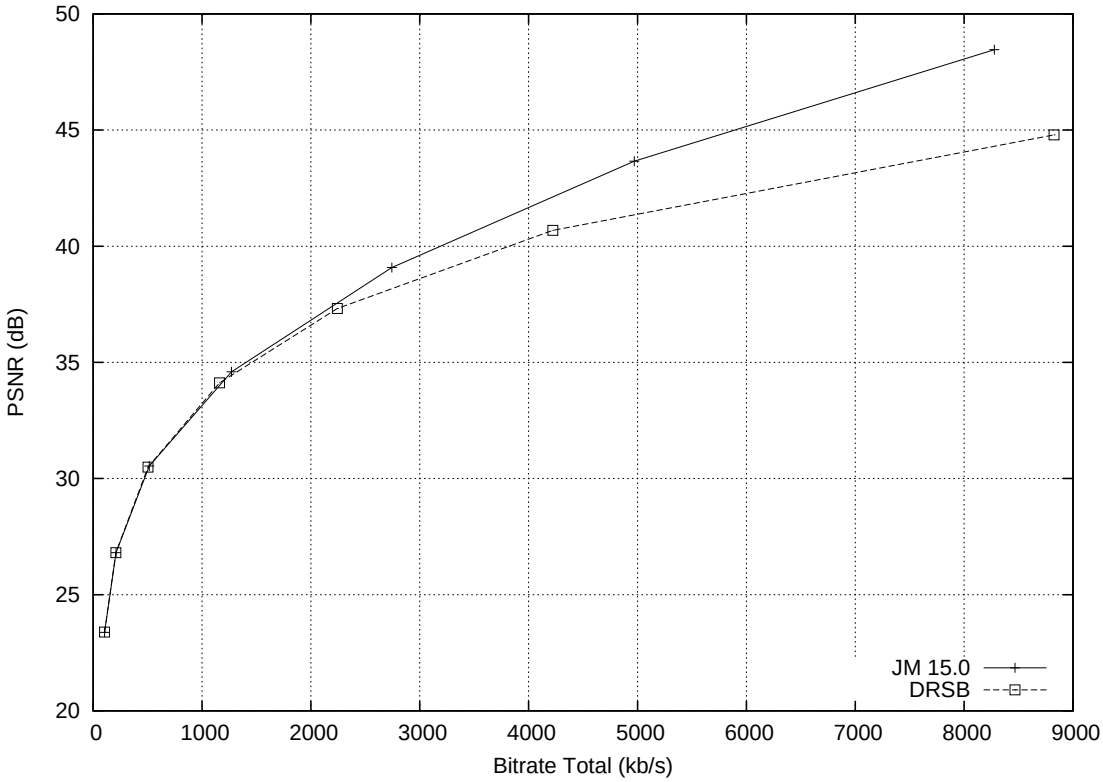


Figura A.12: *Mobile & Calendar* (CIF) - Todos os quadros (Luma)

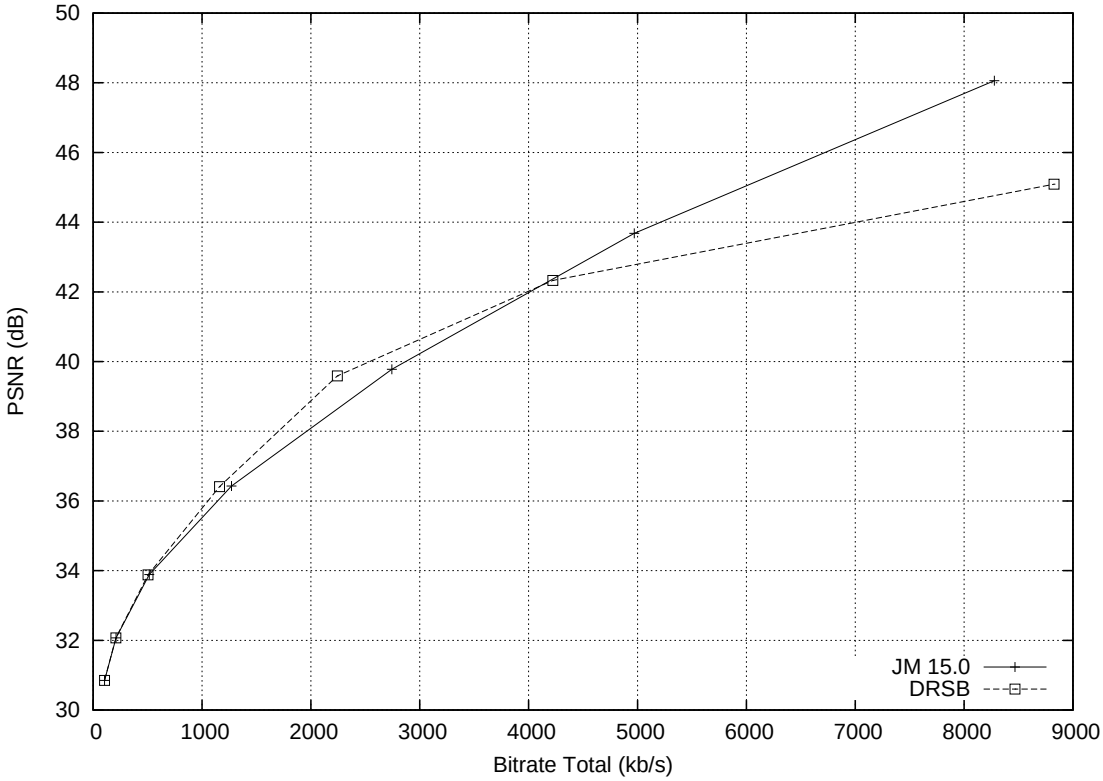


Figura A.13: *Mobile & Calendar* (CIF) - Todos os quadros (Croma - U)

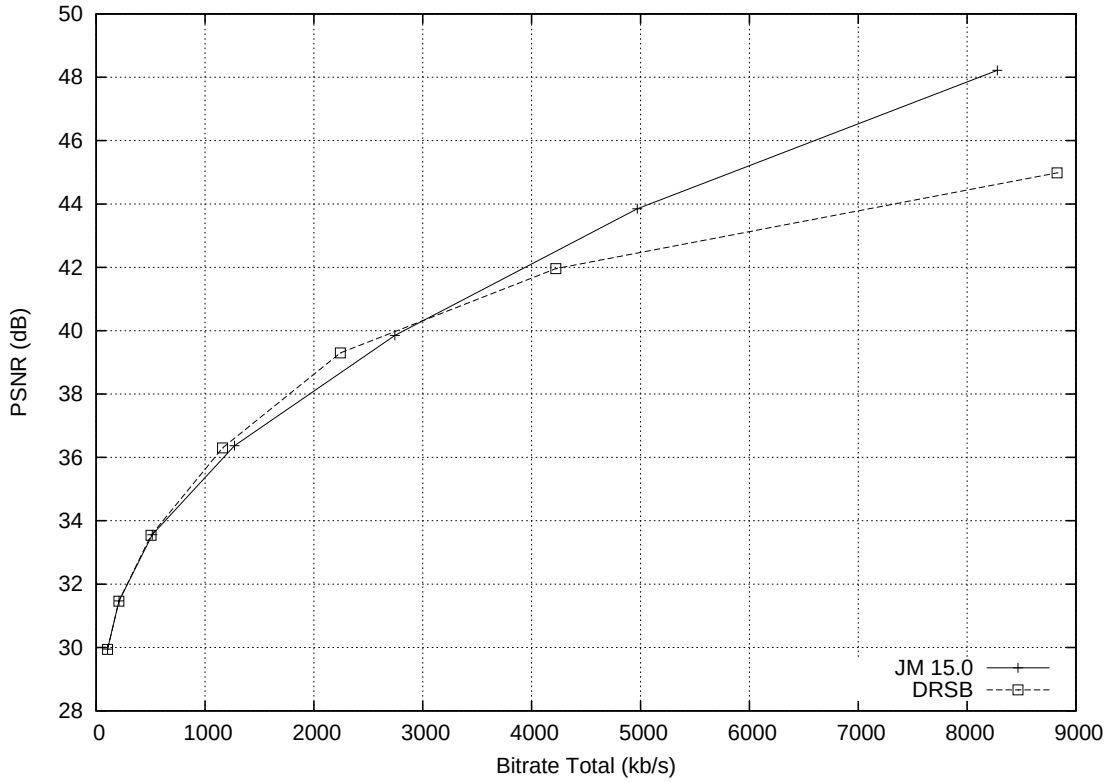


Figura A.14: *Mobile & Calendar* (CIF) - Todos os quadros (Croma - V)

A.1.2 Somentes Quadros com *slices* do Tipo B

Gráficos com informações de todos os quadros contendo *slices* do tipo B de uma mesma sequência para o método DRSB para as componentes de croma.

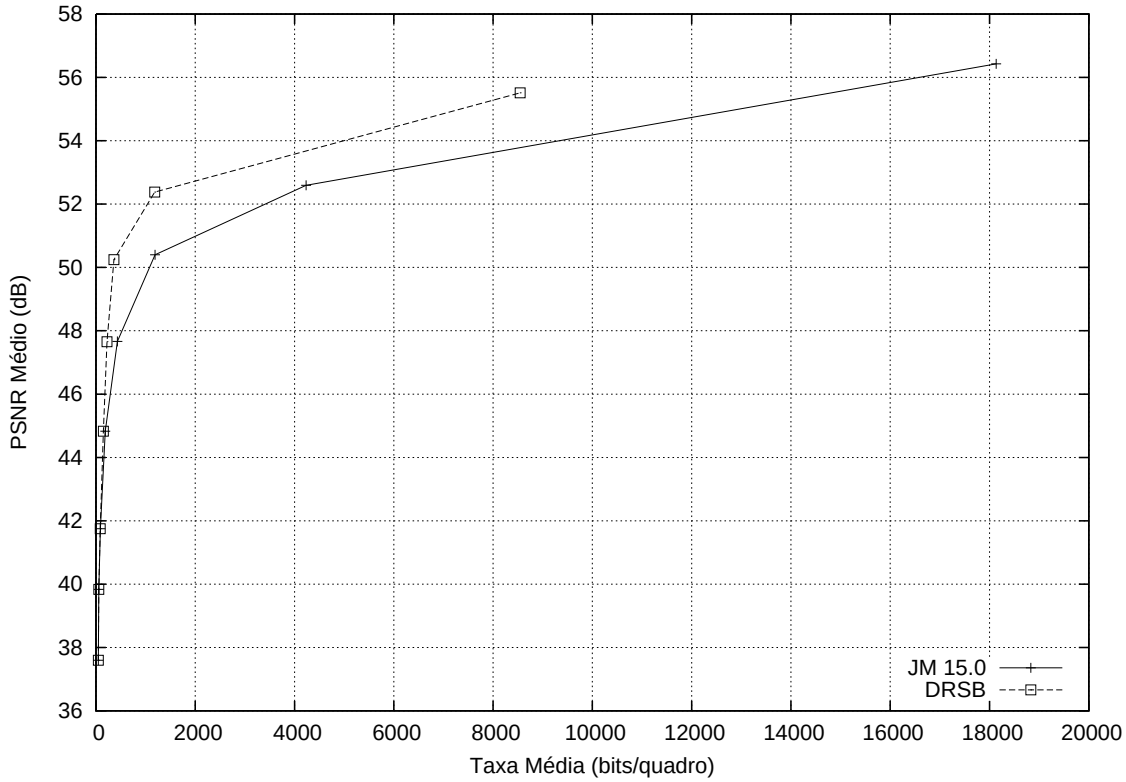


Figura A.16: *Akiyo* (QCIF) - Somente quadros com slices do tipo B (Croma - V)

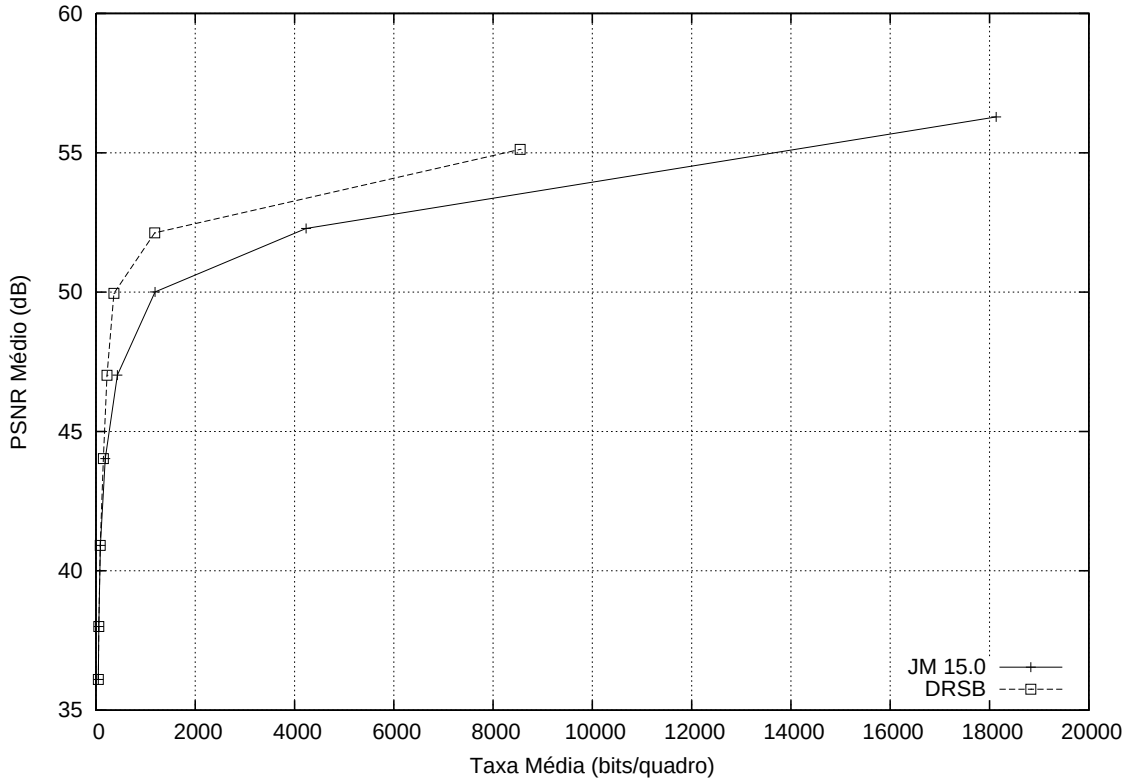


Figura A.15: *Akiyo* (QCIF) - Somente quadros com slices do tipo B (Croma - U)

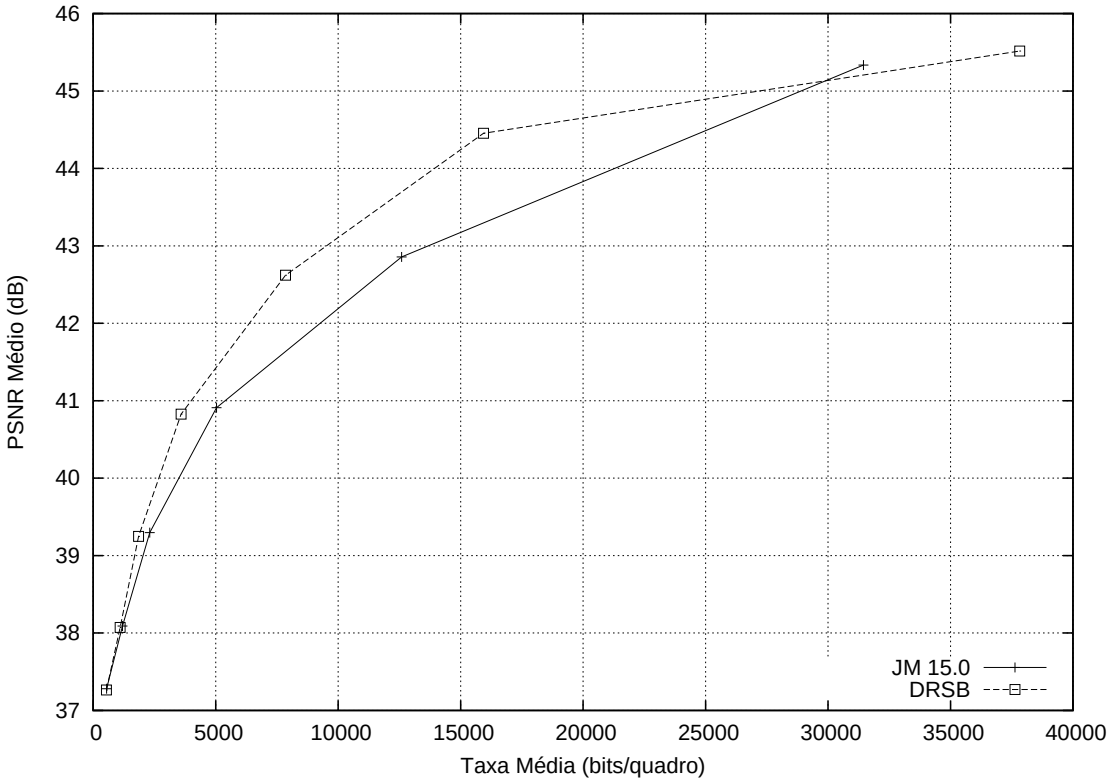


Figura A.17: *Foreman* (CIF) - Somente quadros com slices do tipo B com GOP IBBBP (Croma - U)

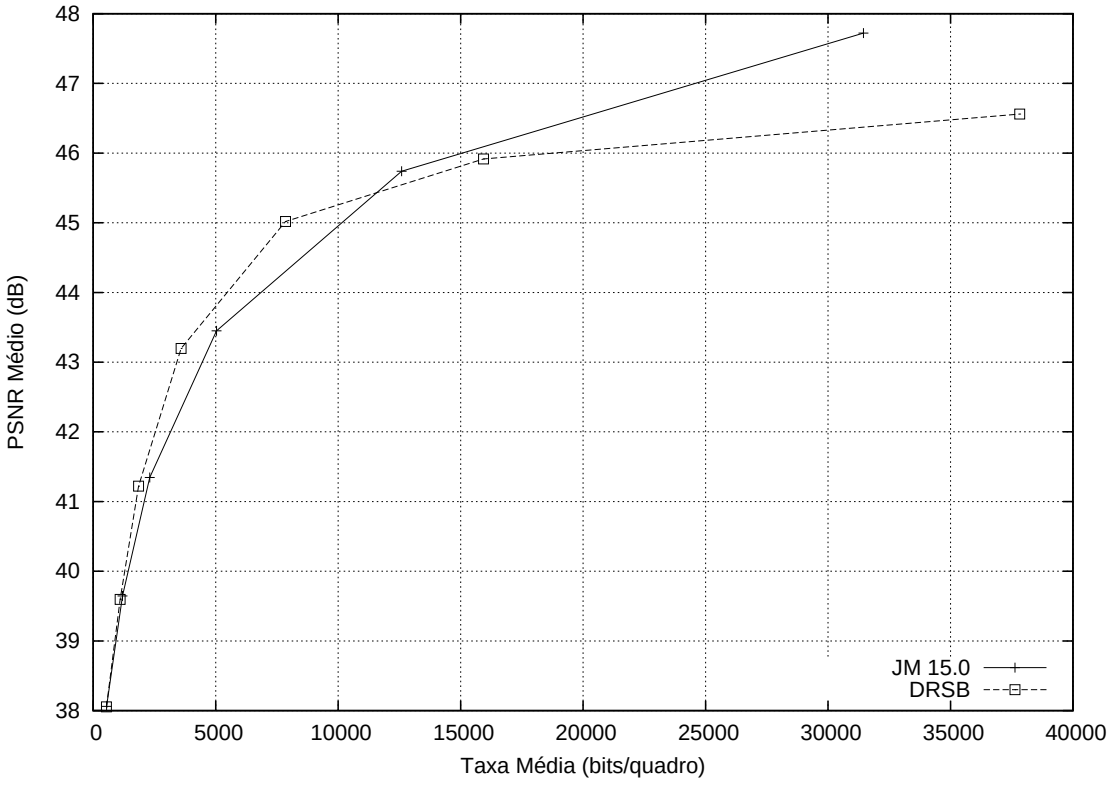


Figura A.18: *Foreman* (CIF) - Somente quadros com slices do tipo B com GOP IBBBP (Croma - V)

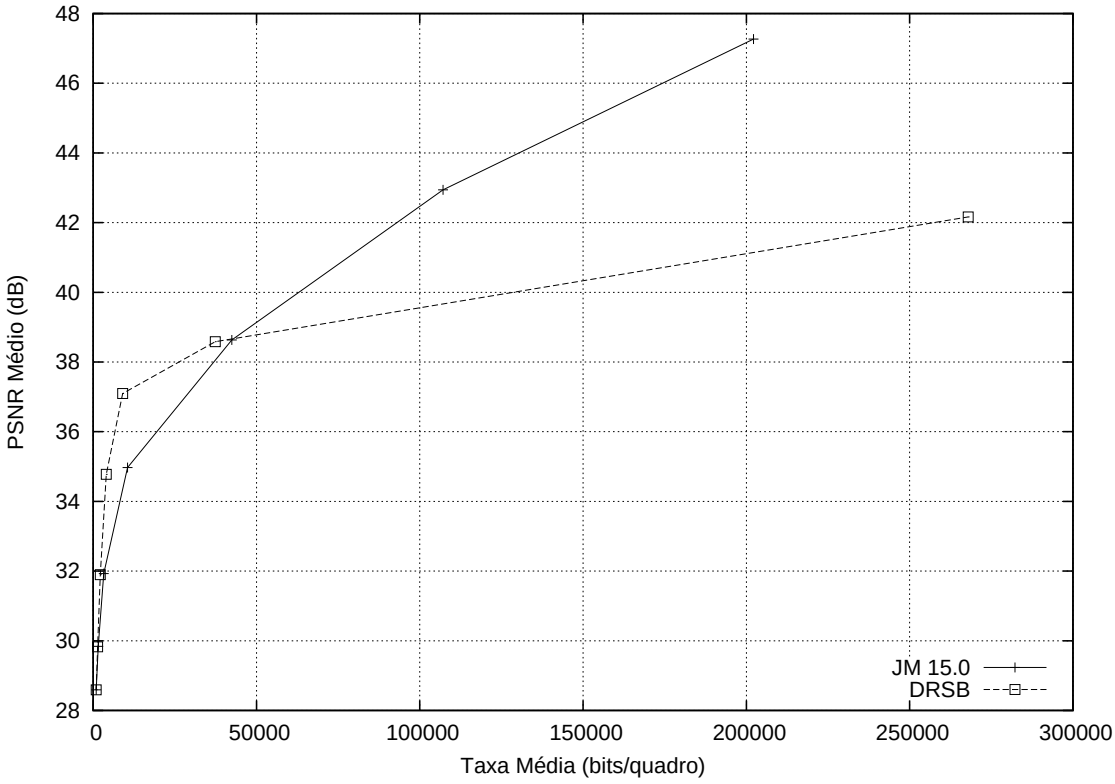


Figura A.19: *Flower Garden* (SIF) - Somente quadros com slices do tipo B (Croma - U)

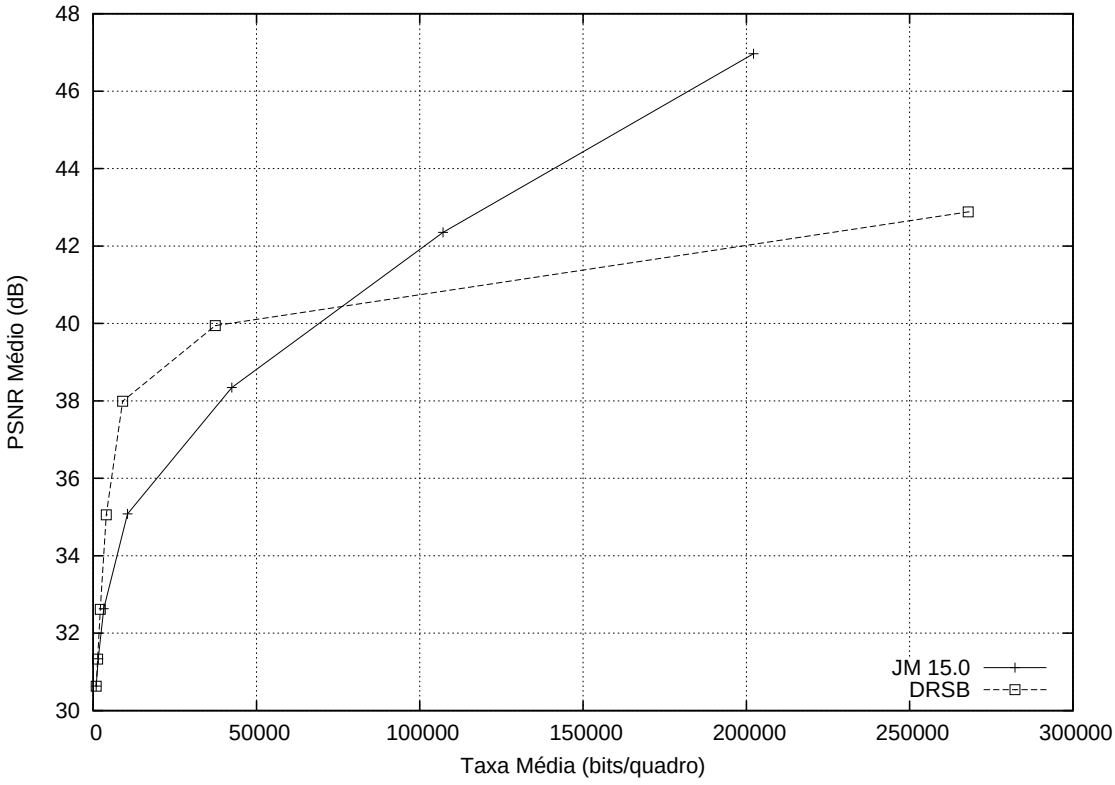


Figura A.20: *Flower Garden* (SIF) - Somente quadros com slices do tipo B (Croma - V)

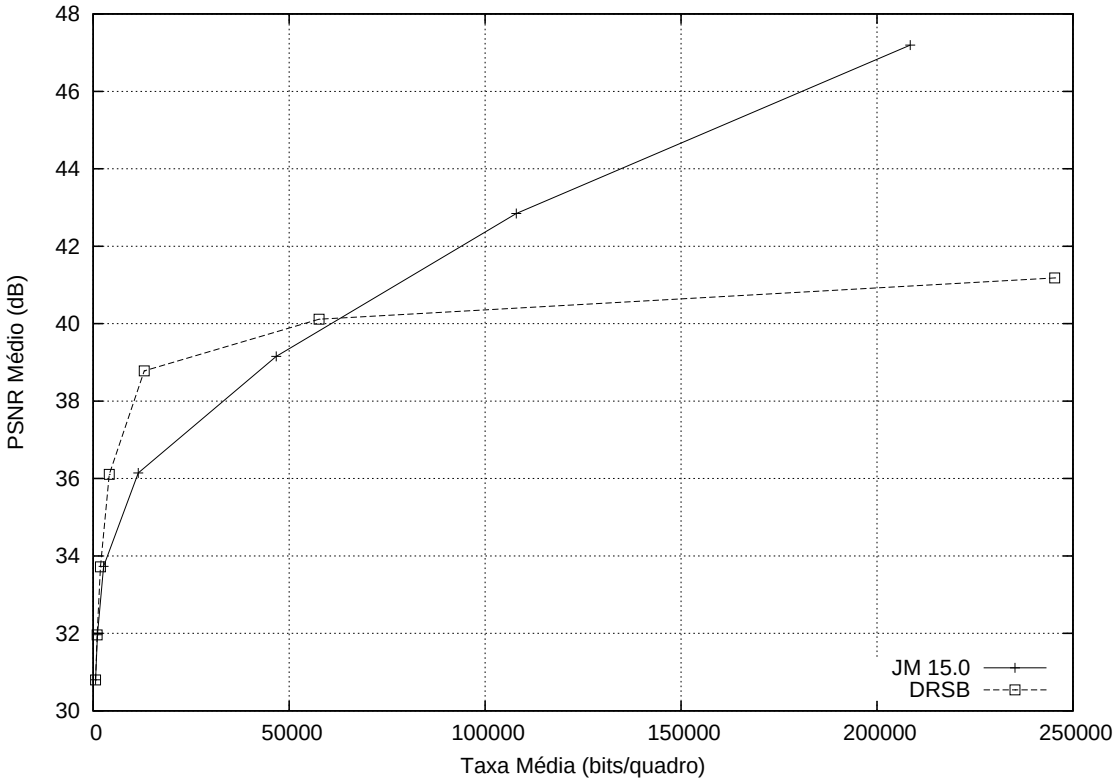


Figura A.21: *Mobile & Calendar* (CIF) - Somente quadros com slices do tipo B (Croma - U)

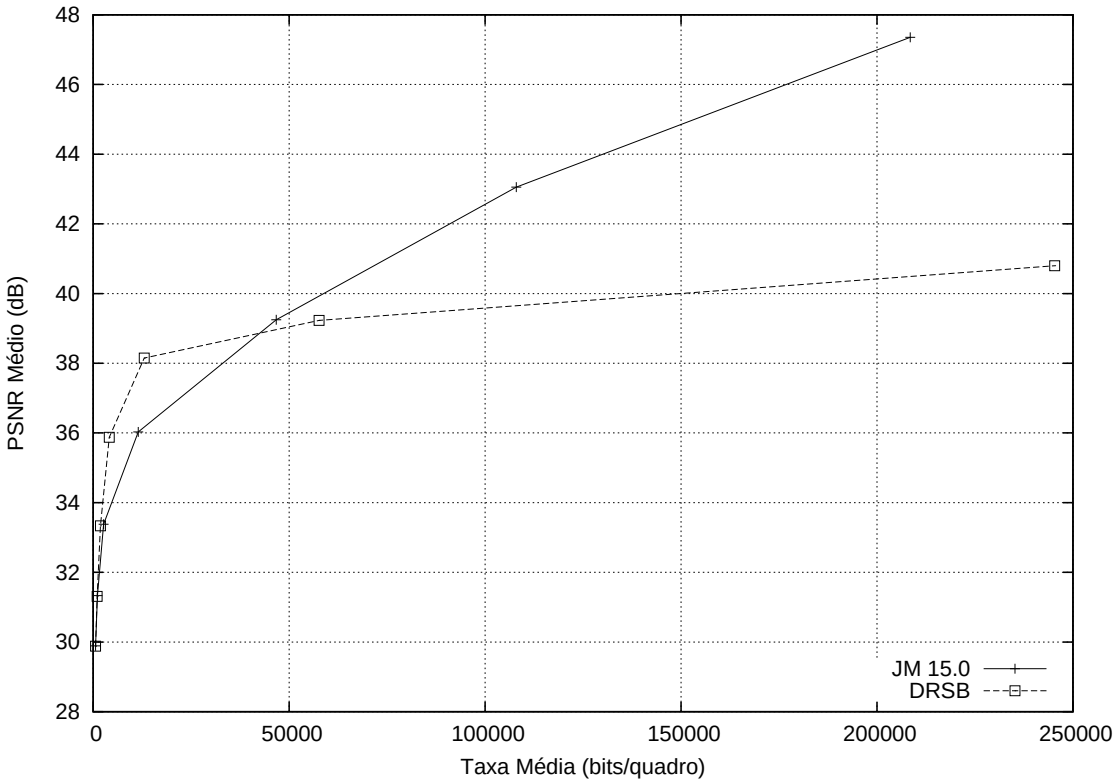


Figura A.22: *Mobile & Calendar* (CIF) - Somente quadros com slices do tipo B (Croma - V)

A.2 OCSB e DRSB

A.2.1 Todos os Quadros

Gráficos contendo informações de todos os quadros de uma mesma sequência para os métodos OCSB e DRSB.

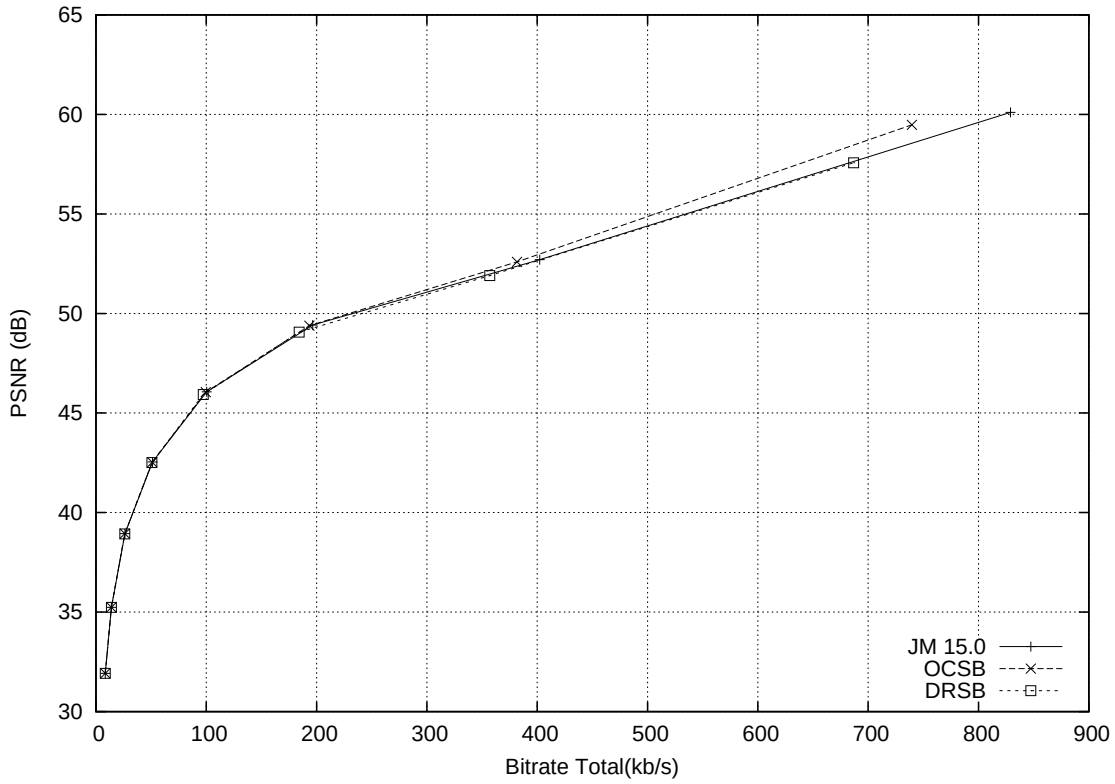


Figura A.23: *Akiyo* (QCIF) - Todos os quadros (Luma)

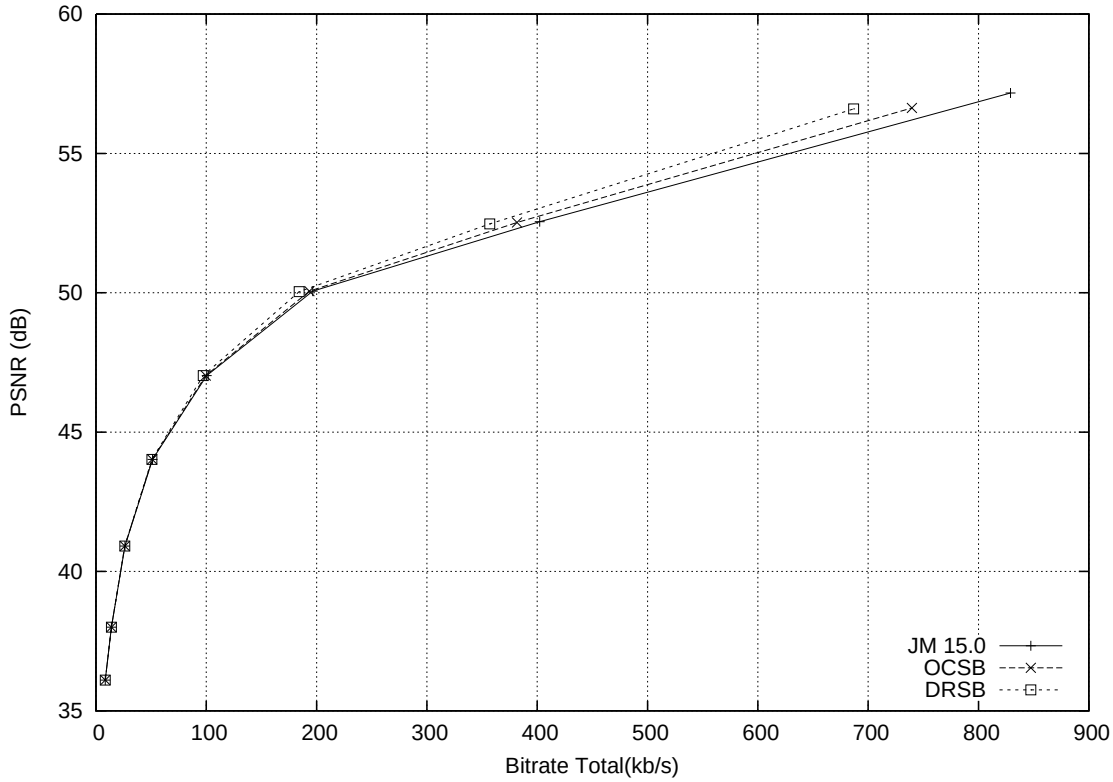


Figura A.24: *Akiyo* (QCIF) - Todos os quadros (Croma - U)

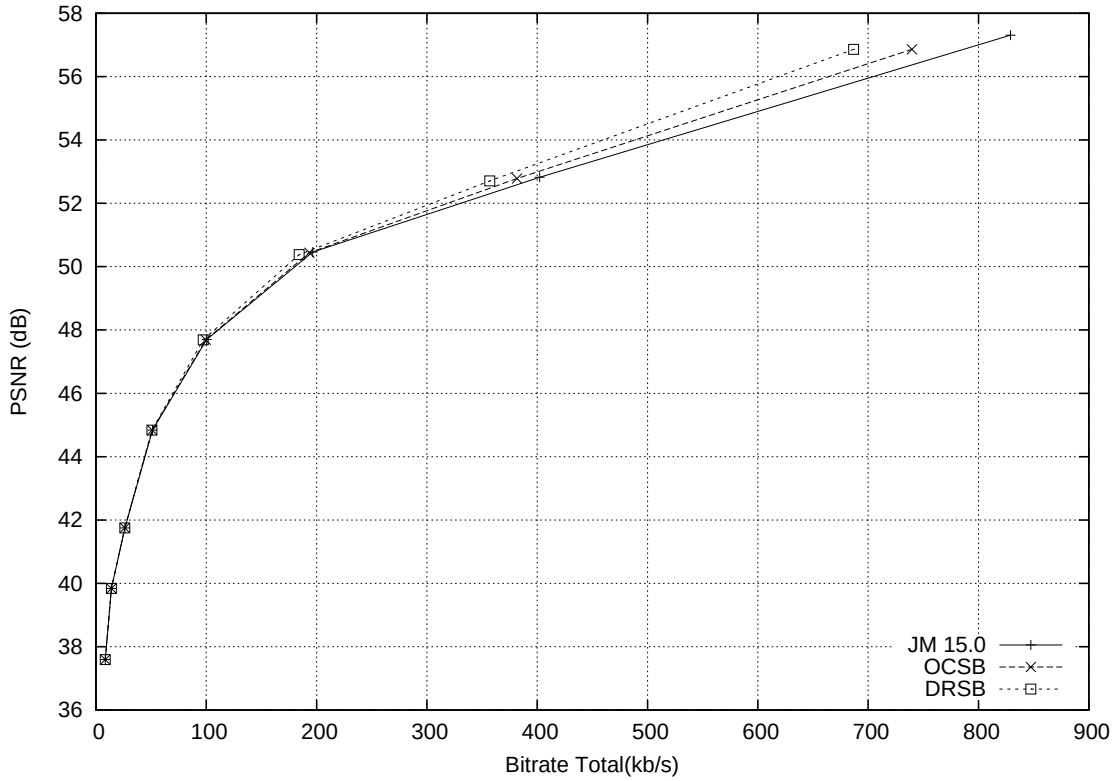


Figura A.25: *Akiyo* (QCIF) - Todos os quadros (Croma - V)

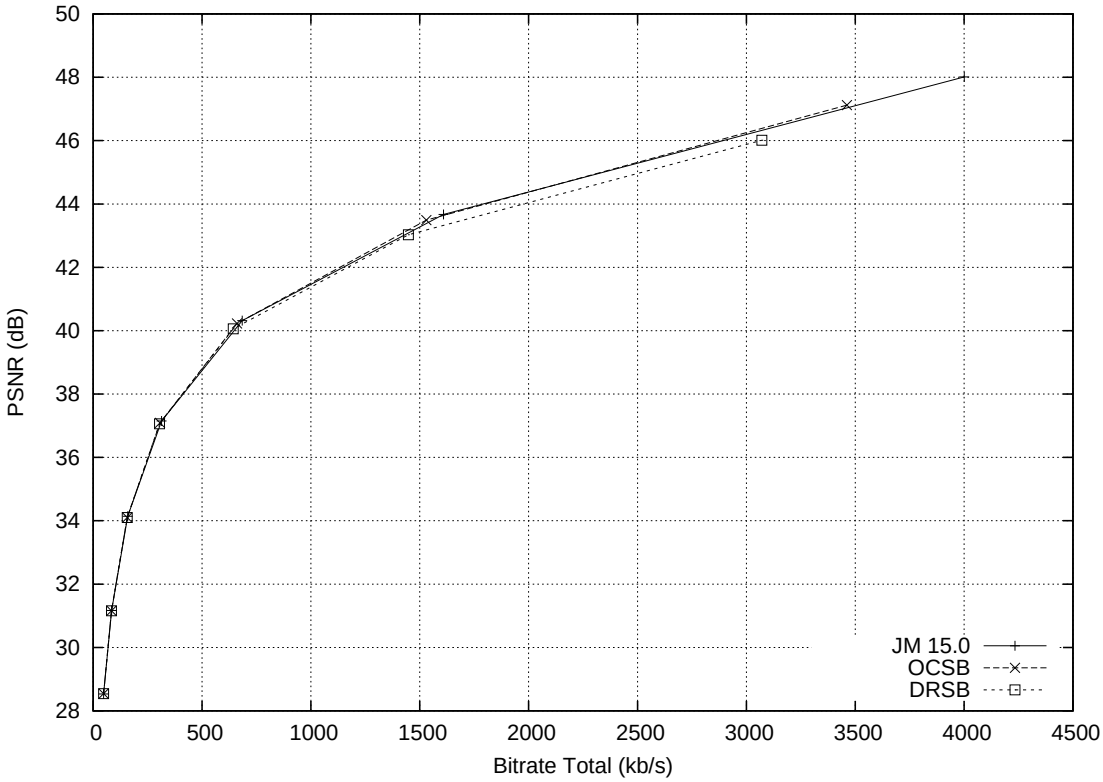


Figura A.26: *Foreman* (CIF) - Todos os quadros (Luma)

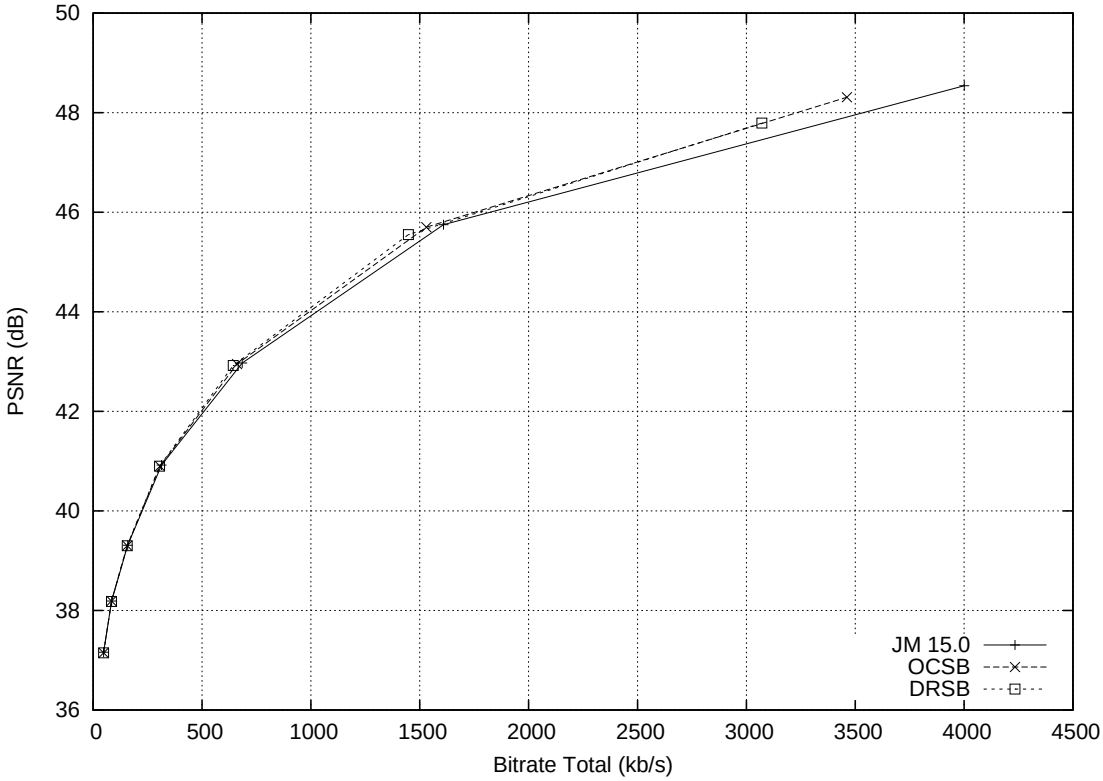


Figura A.27: *Foreman* (CIF) - Todos os quadros (Croma - U)

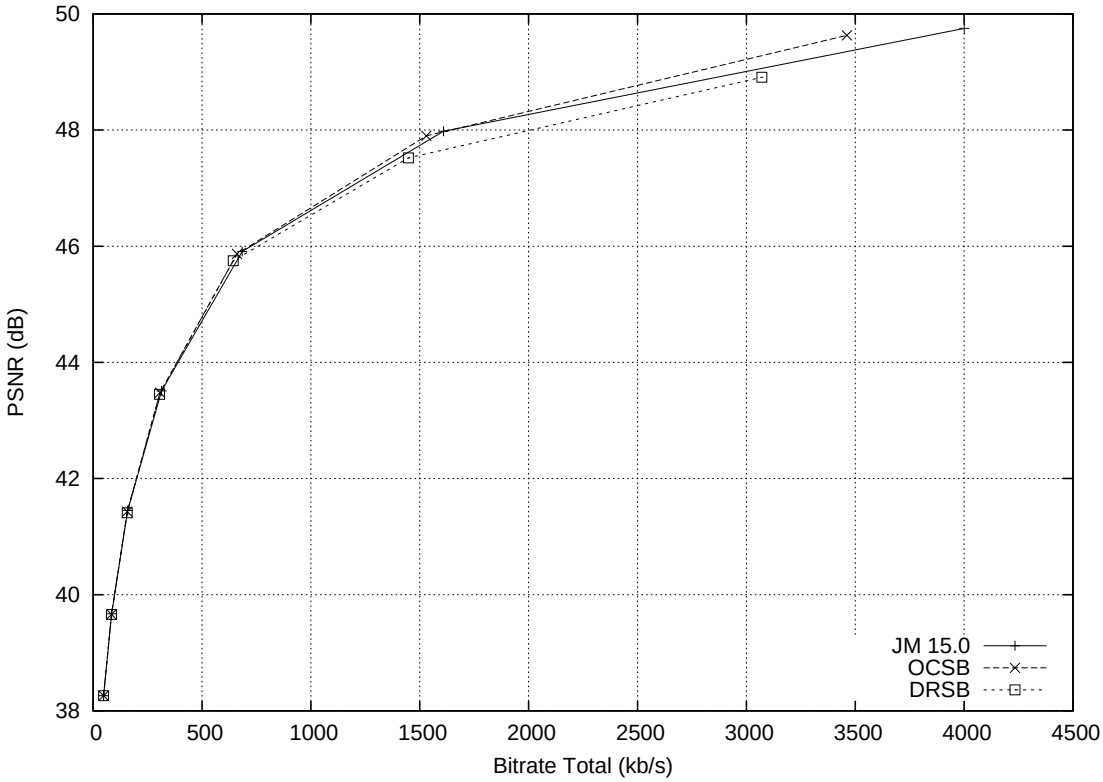


Figura A.28: *Foreman* (CIF) - Todos os quadros (Croma - V)

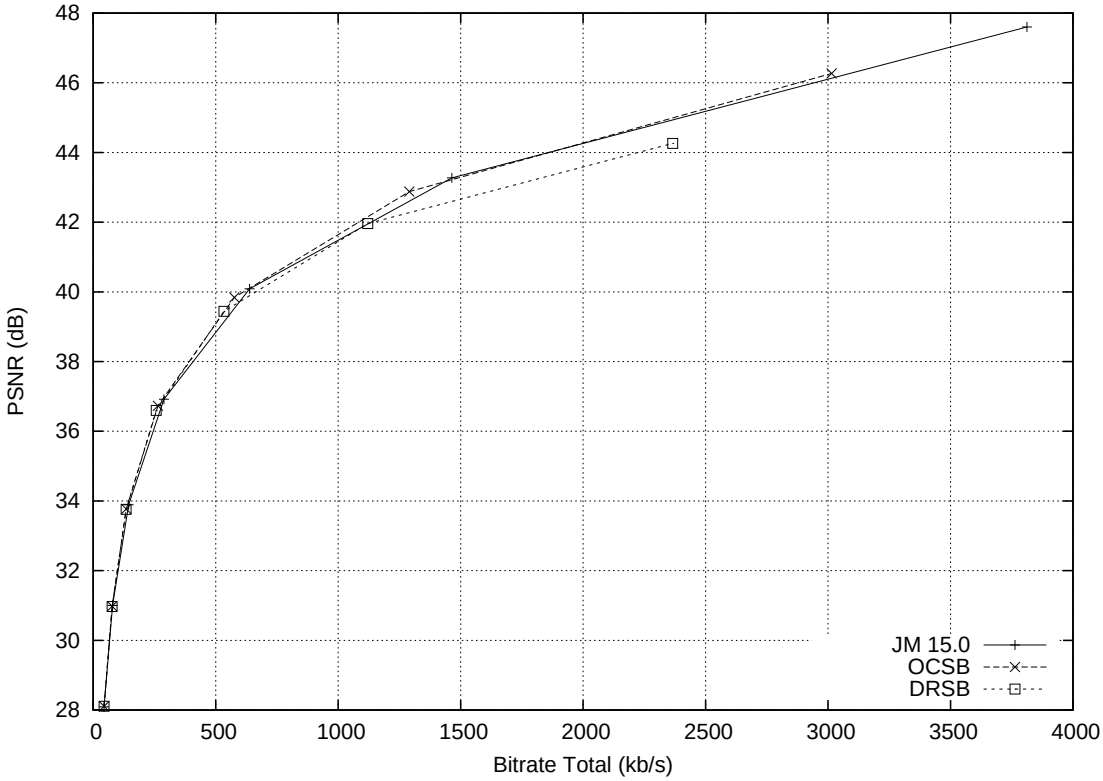


Figura A.29: *Foreman* (CIF) - Todos os quadros com GOP IBBBP (Luma)

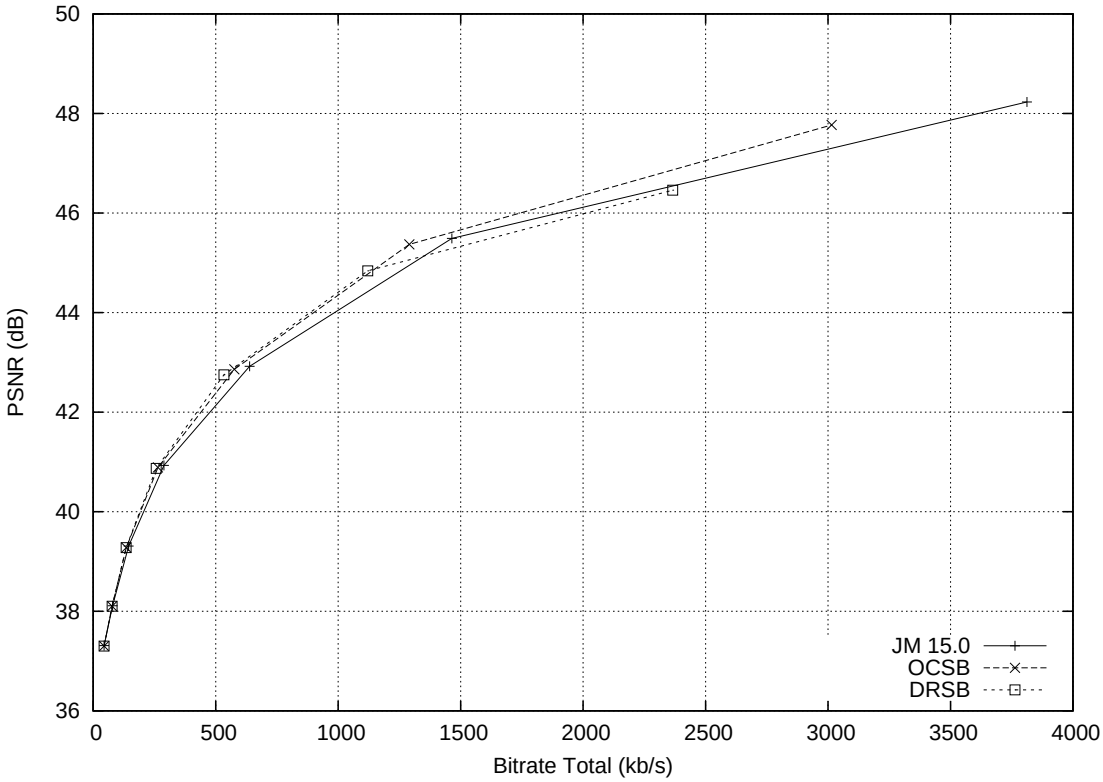


Figura A.30: *Foreman* (CIF) - Todos os quadros com GOP IBBBP (Croma - U)

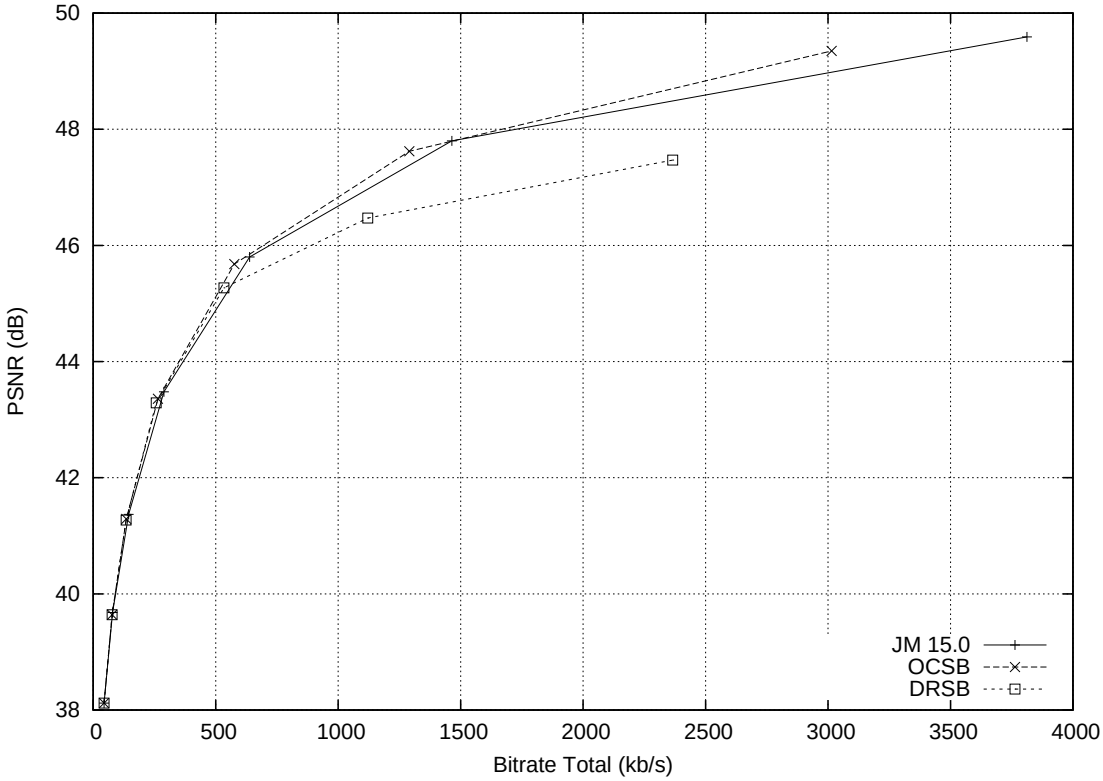


Figura A.31: *Foreman* (CIF) - Todos os quadros com GOP IBBBP (Croma - V)

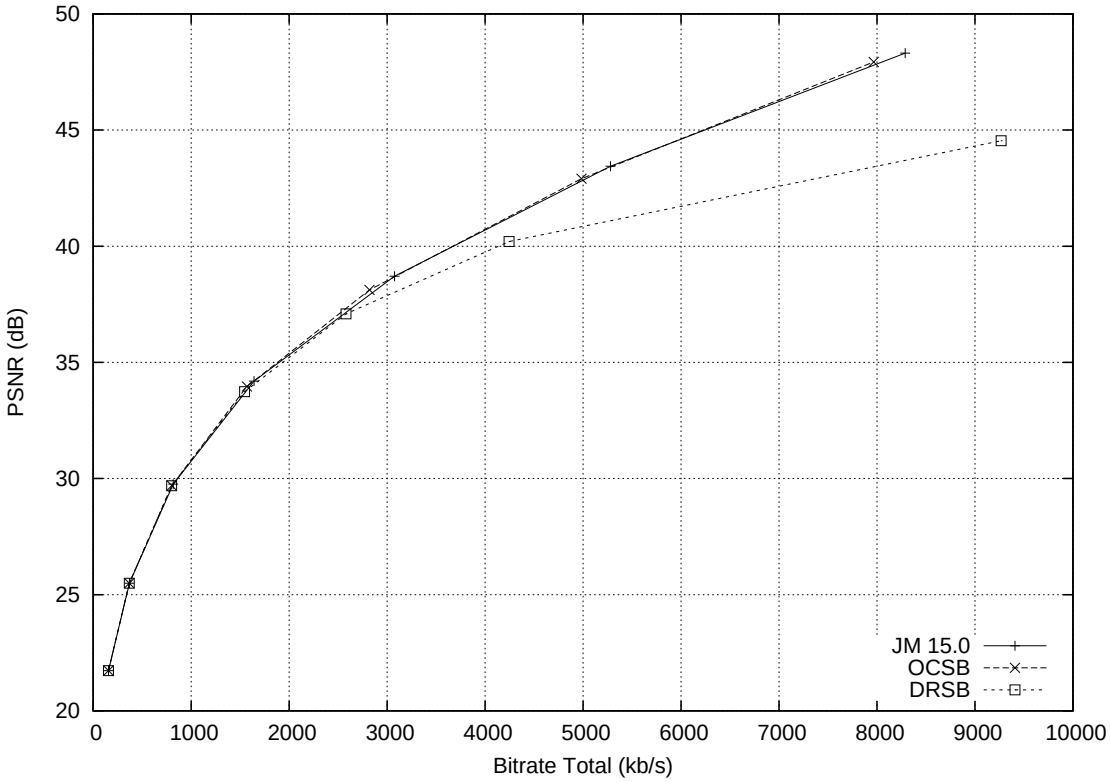


Figura A.32: *Flower Garden* (SIF) - Todos os quadros (Luma)

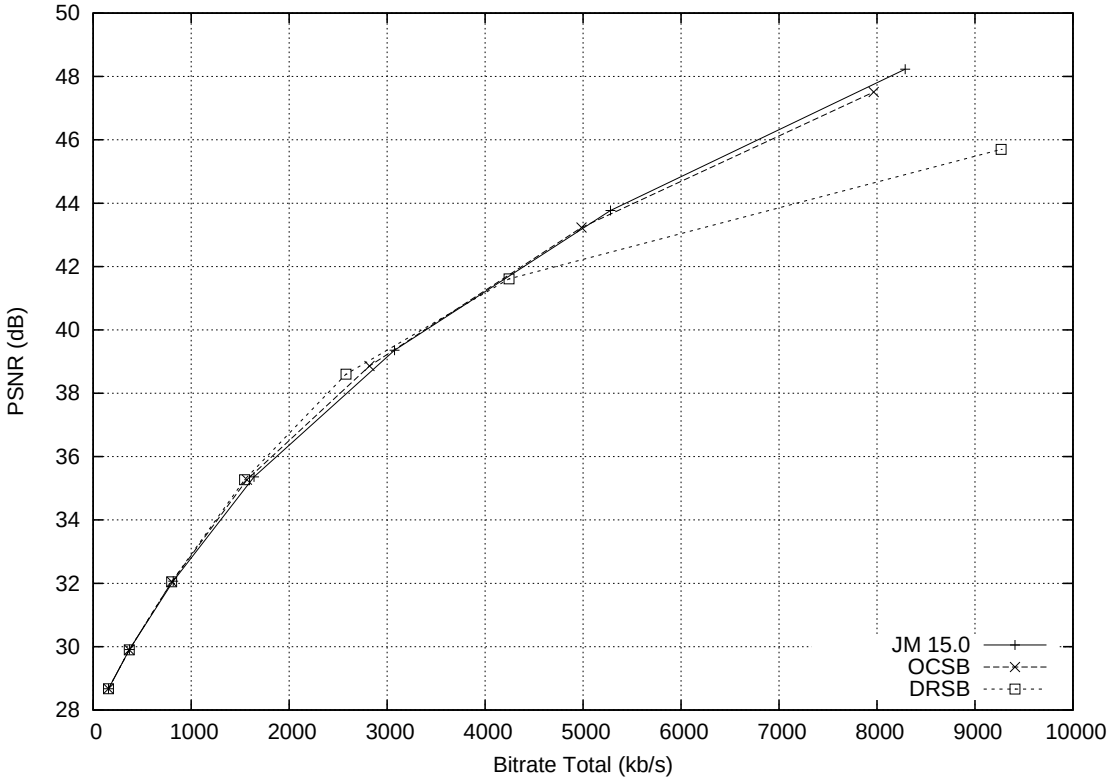


Figura A.33: *Flower Garden* (SIF) - Todos os quadros (Croma - U)

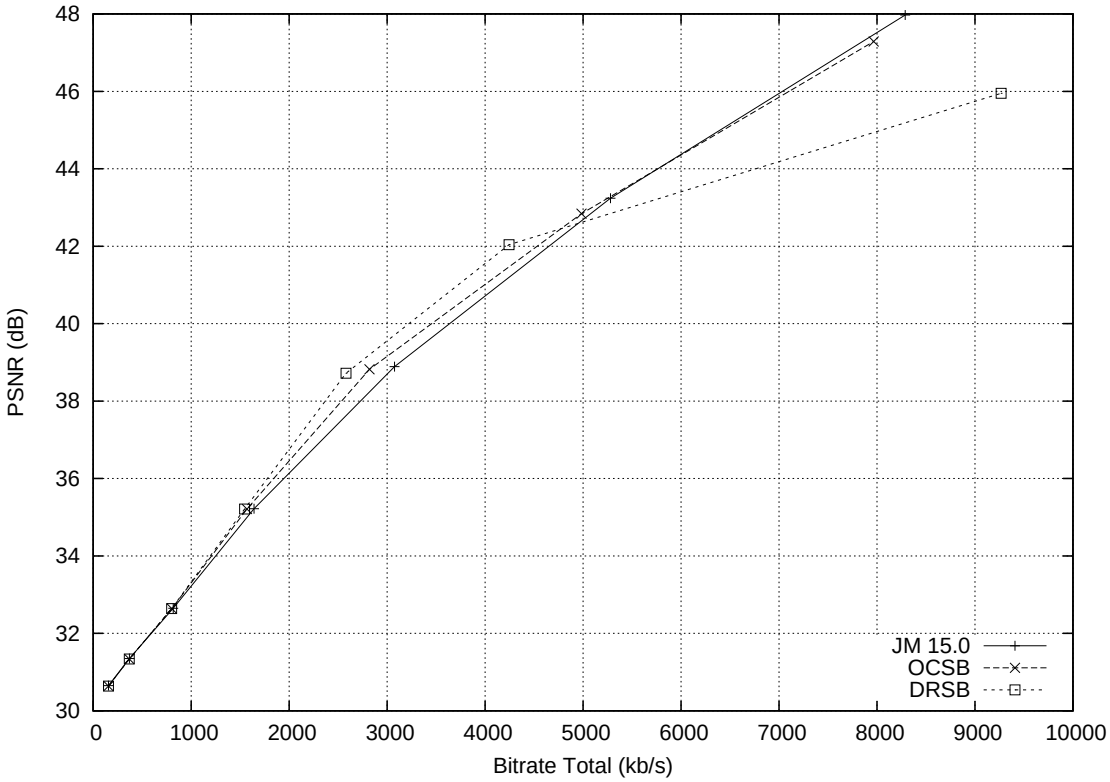


Figura A.34: *Flower Garden* (SIF) - Todos os quadros (Croma - V)

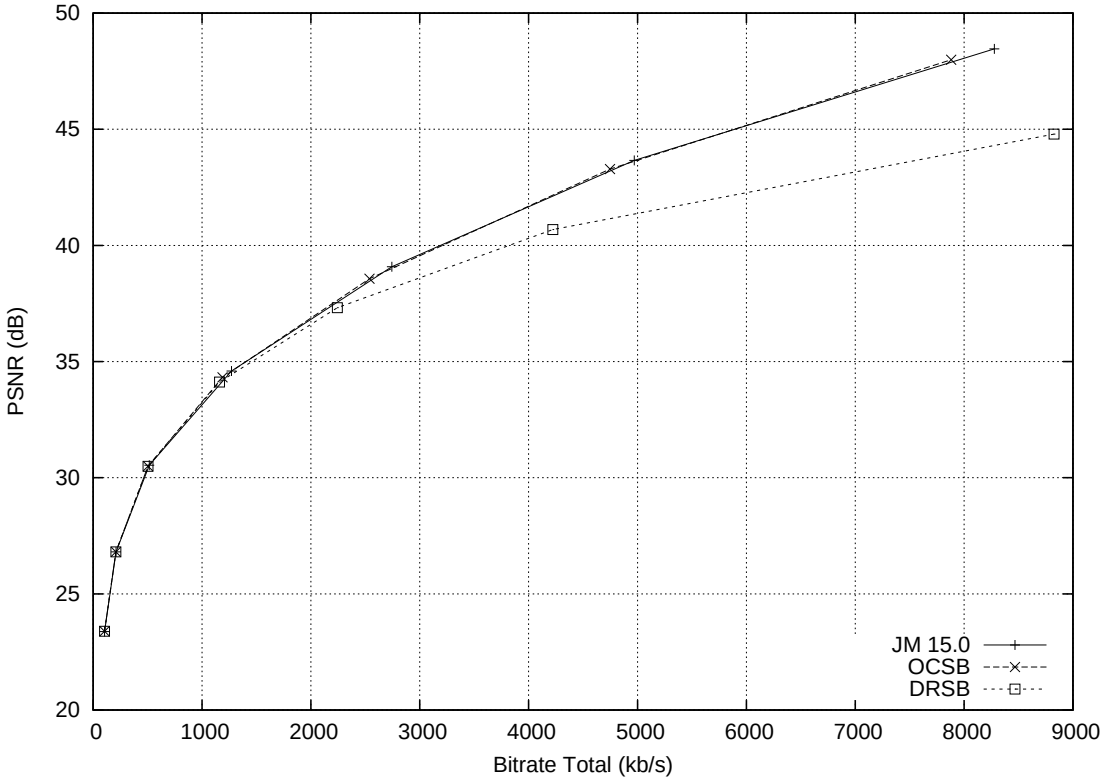


Figura A.35: *Mobile & Calendar* (CIF) - Todos os quadros (Luma)

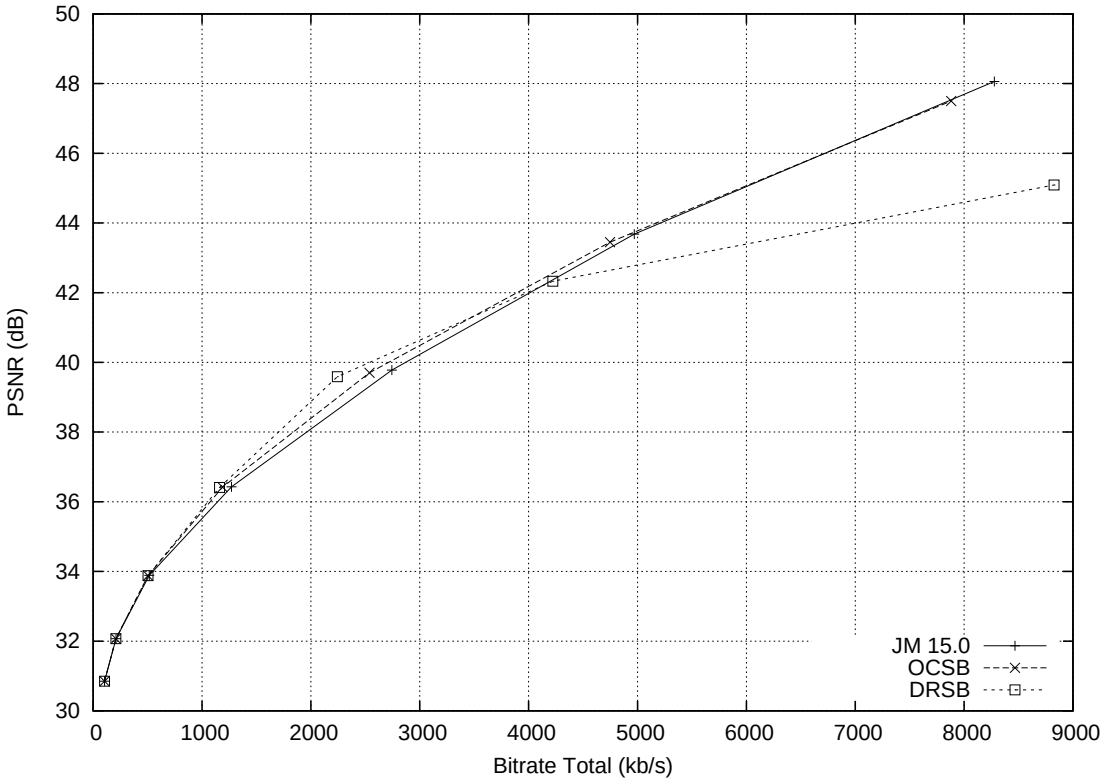


Figura A.36: *Mobile & Calendar* (CIF) - Todos os quadros (Croma - U)

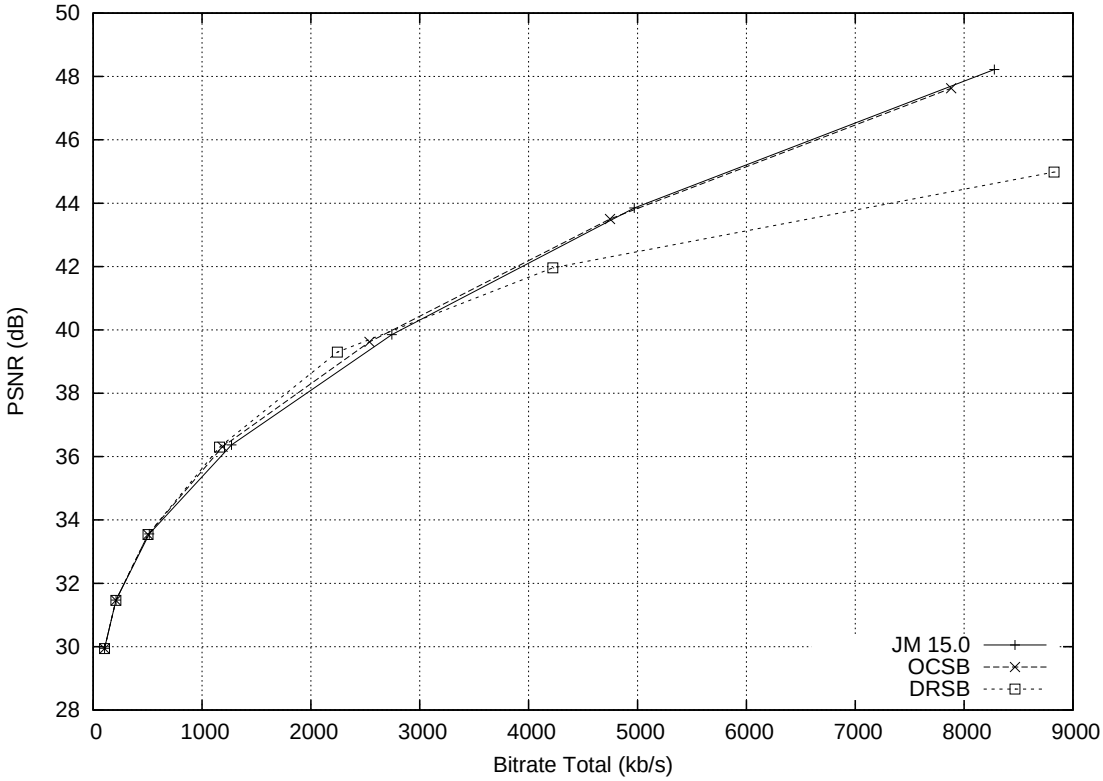


Figura A.37: *Mobile & Calendar* (CIF) - Todos os quadros (Croma - V)

A.2.2 Somentes Quadros com *slices* do Tipo B

Gráficos com informações todos os quadros contendo *slices* do tipo B de uma mesma sequência para os métodos OCSB e DRSB para as componentes de croma.

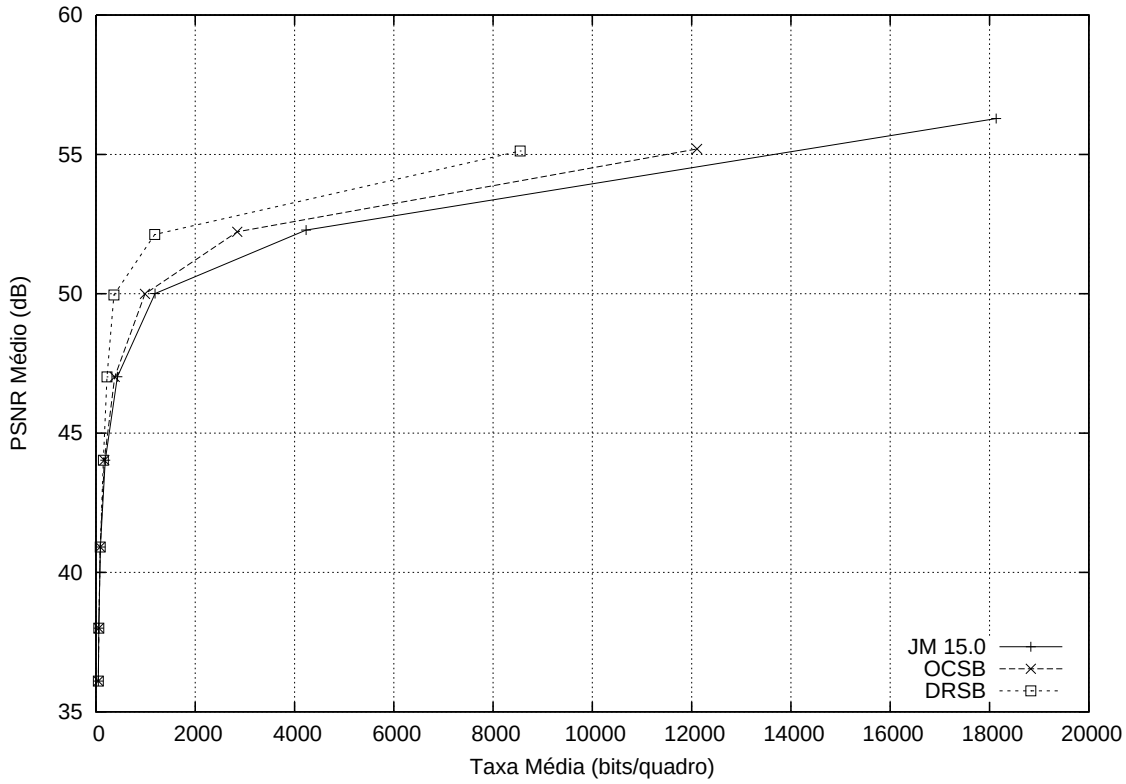


Figura A.38: *Akiyo* (QCIF) - Somente quadros com slices do tipo B (Croma - U)

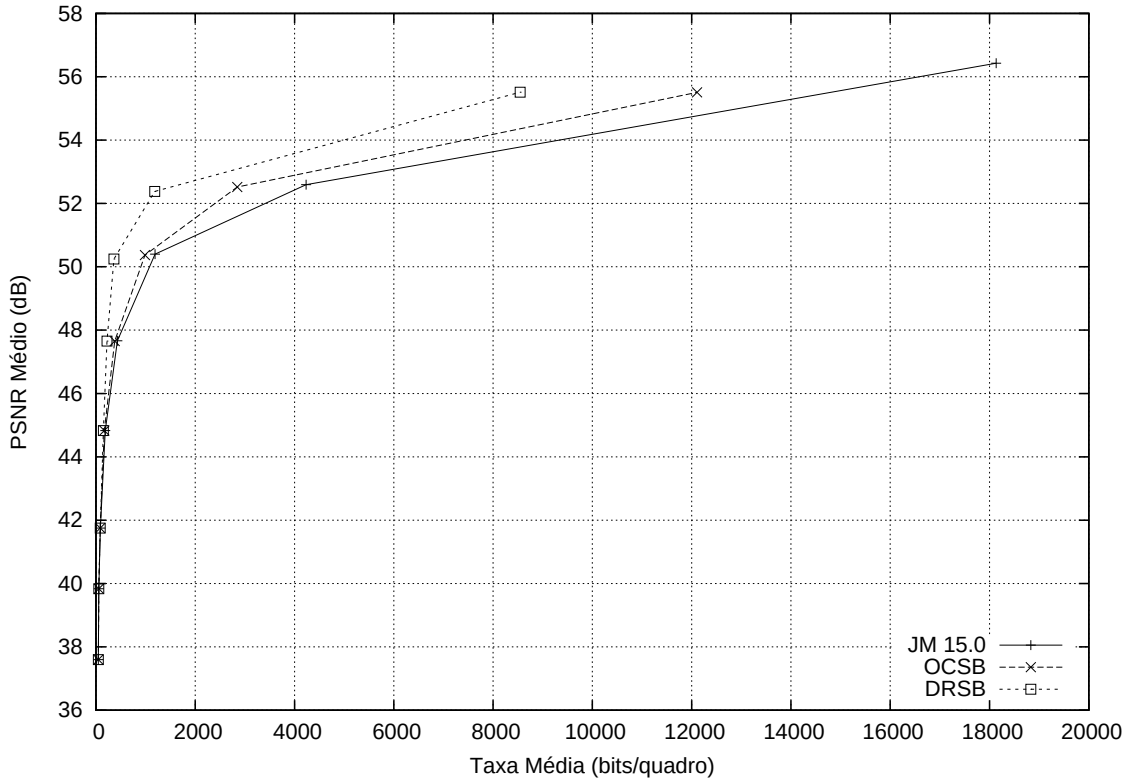


Figura A.39: *Akiyo* (QCIF) - Somente quadros com slices do tipo B (Croma - V)

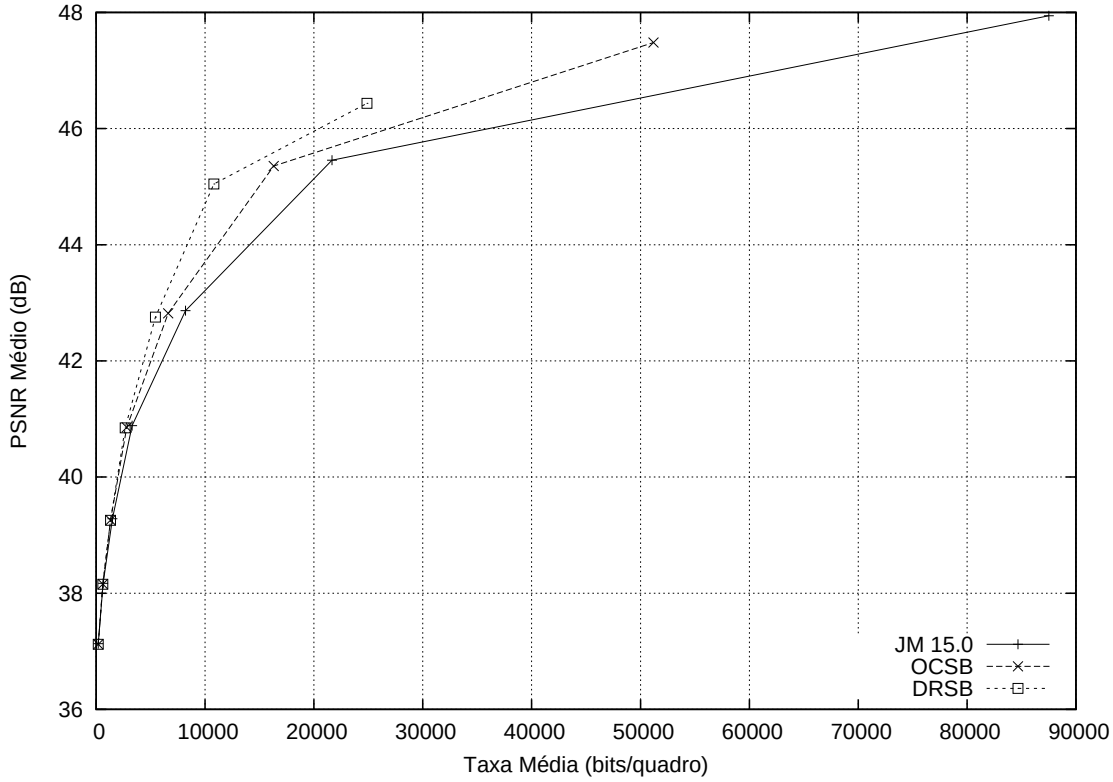


Figura A.40: *Foreman* (CIF) - Somente quadros com slices do tipo B (Croma - U)

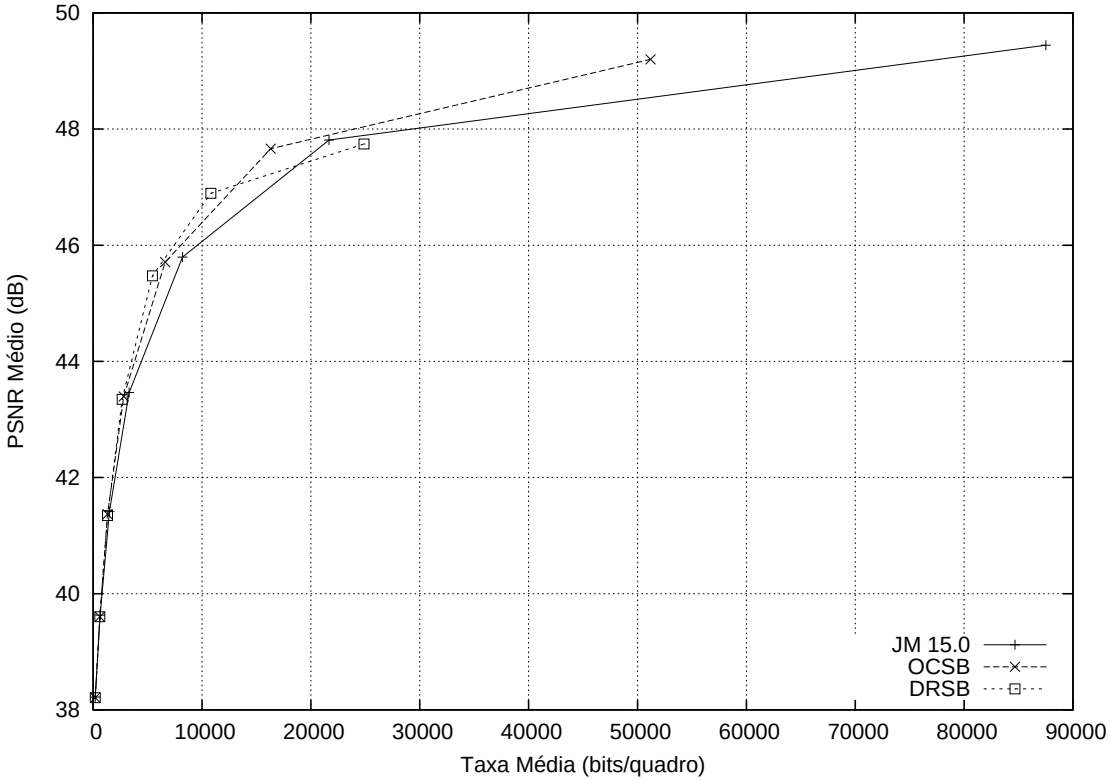


Figura A.41: *Foreman* (CIF) - Somente quadros com slices do tipo B (Croma - V)

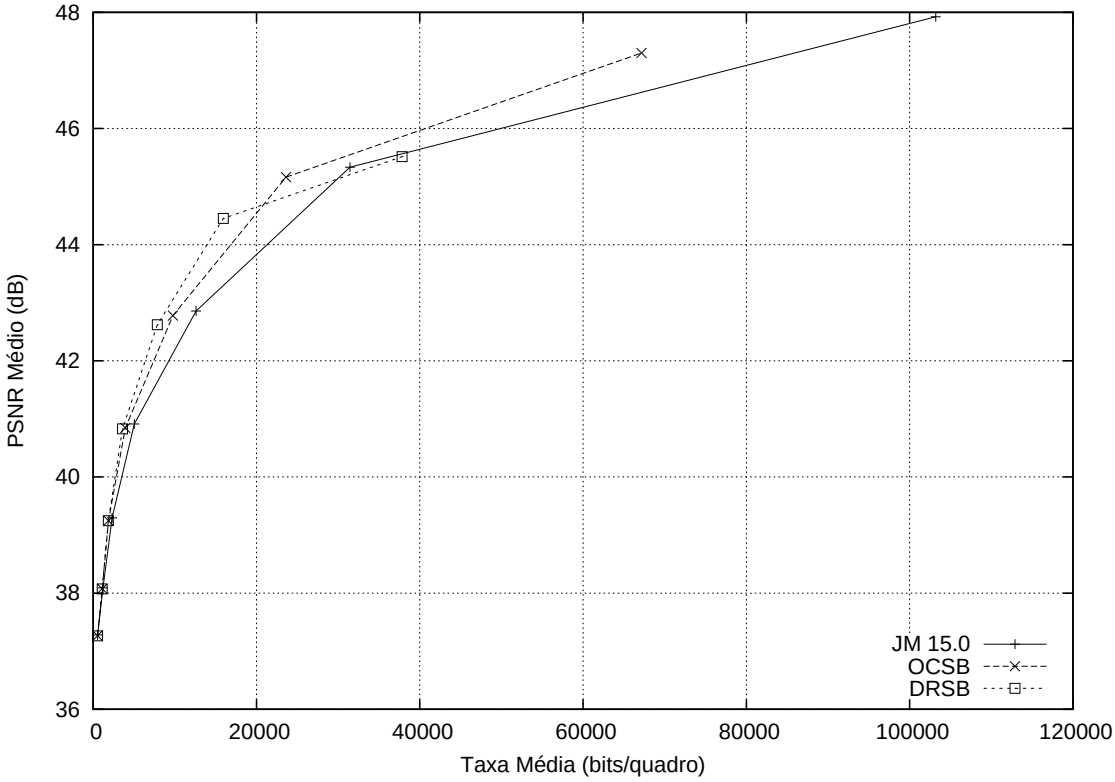


Figura A.42: *Foreman* (CIF) - Somente quadros com slices do tipo B com GOP IBBBP (Croma - U)

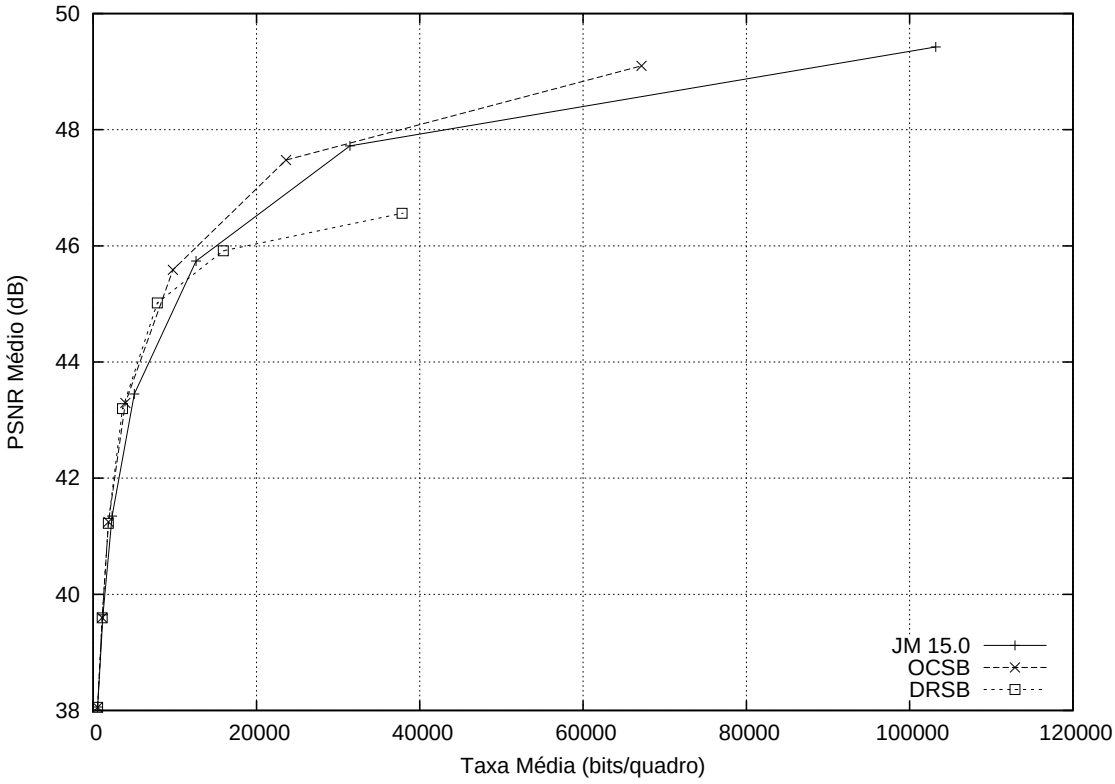


Figura A.43: *Foreman* (CIF) - Somente quadros com slices do tipo B com GOP IBBBP (Croma - V)

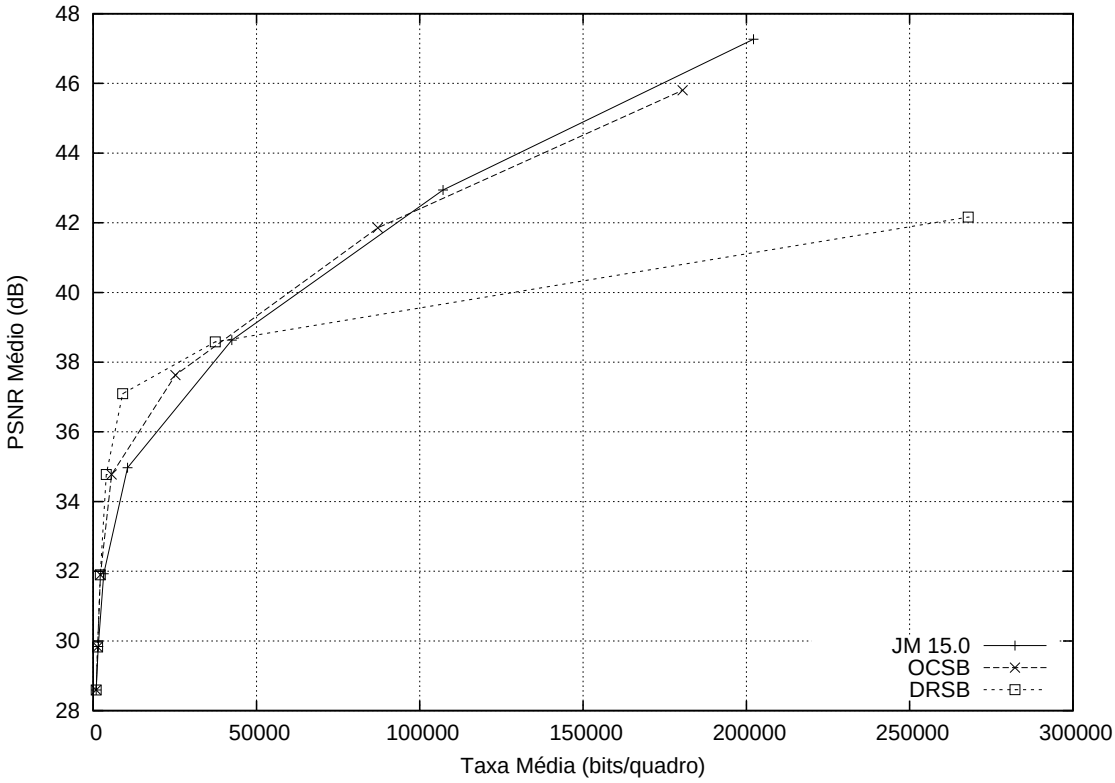


Figura A.44: *Flower Garden* (SIF) - Somente quadros com slices do tipo B (Croma - U)

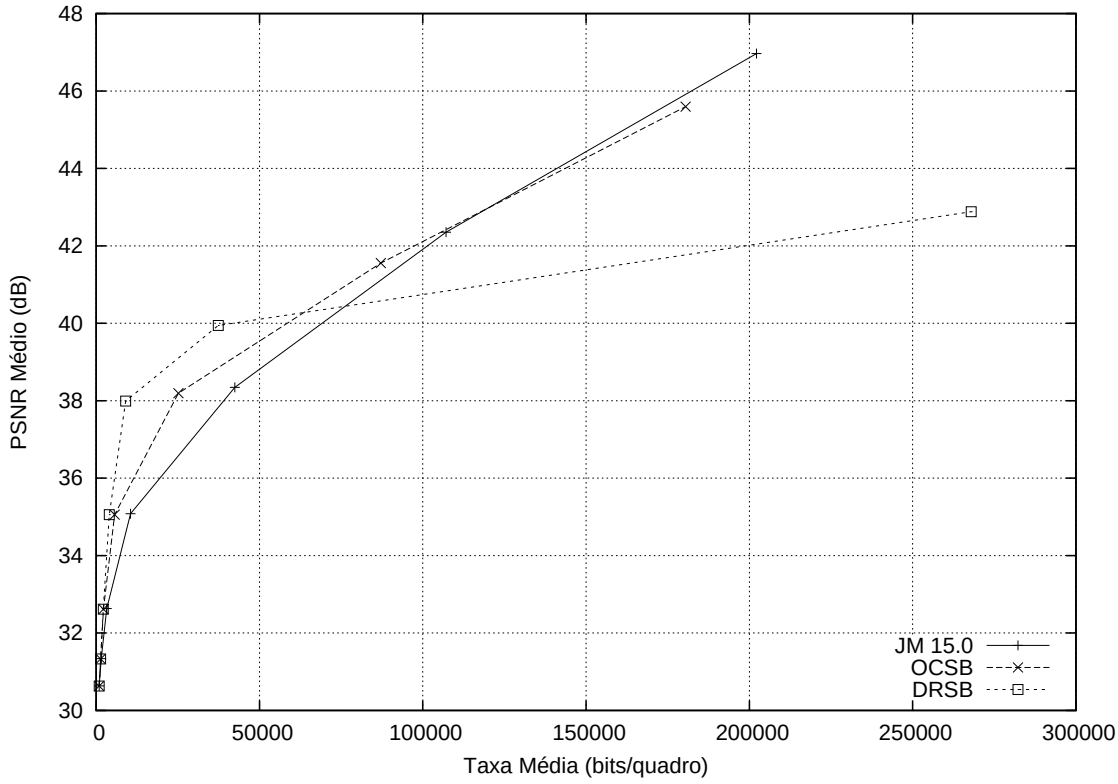


Figura A.45: *Flower Garden* (SIF) - Somente quadros com slices do tipo B (Croma - V)

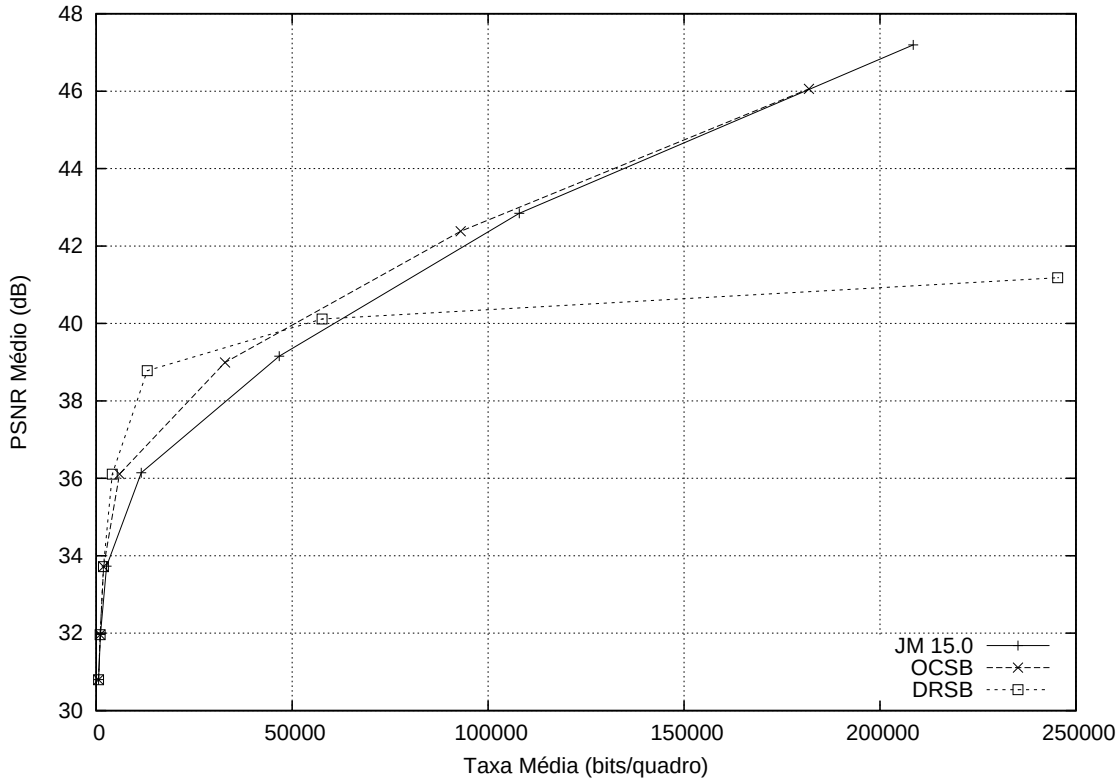


Figura A.46: *Mobile & Calendar* (CIF) - Somente quadros com slices do tipo B (Croma - U)

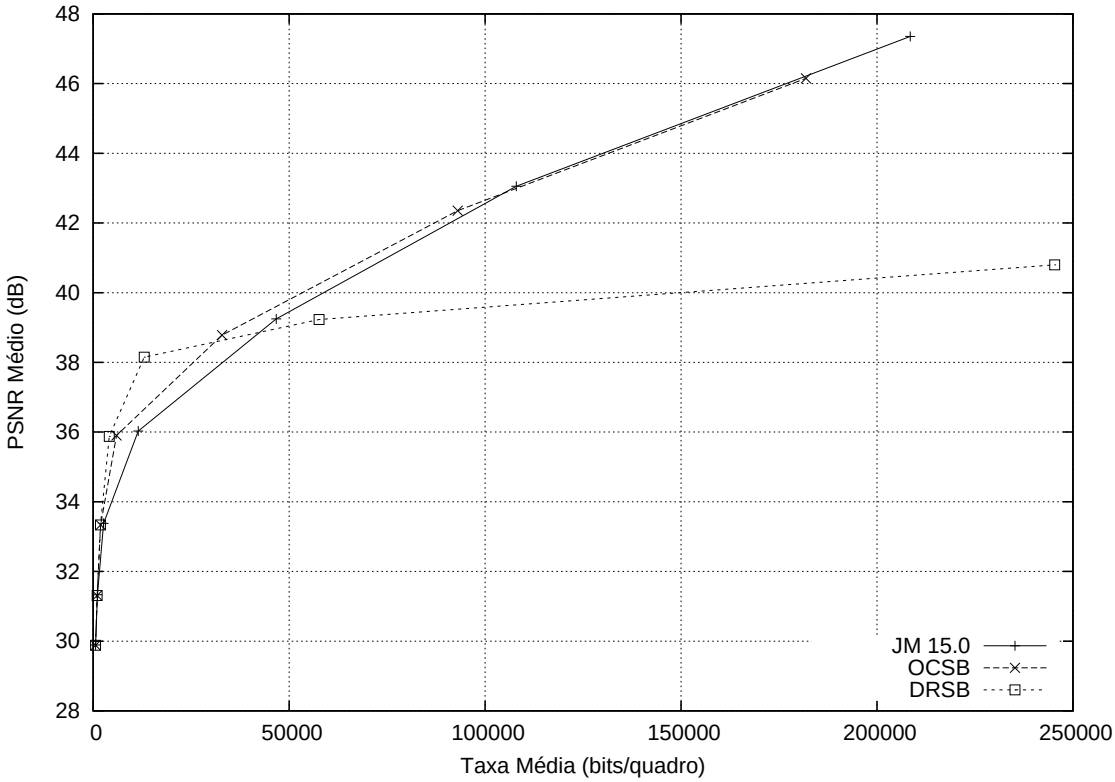


Figura A.47: *Mobile & Calendar* (CIF) - Somente quadros com slices do tipo B (Croma - V)

A.3 CPST e DRSB16.2

A.3.1 Todos os Quadros

Gráficos contendo informações de todos os quadros de uma mesma sequência para os métodos CPST e DRSB16.2.

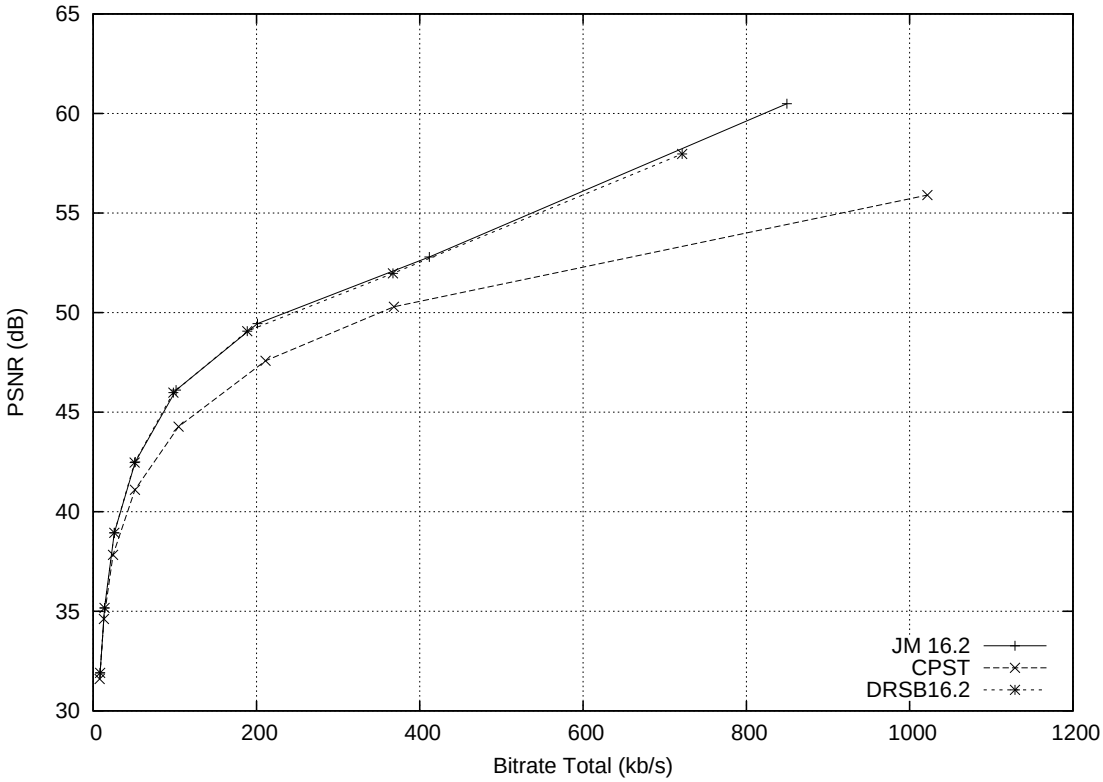


Figura A.48: *Akiyo* (QCIF) - Todos os quadros (Luma)

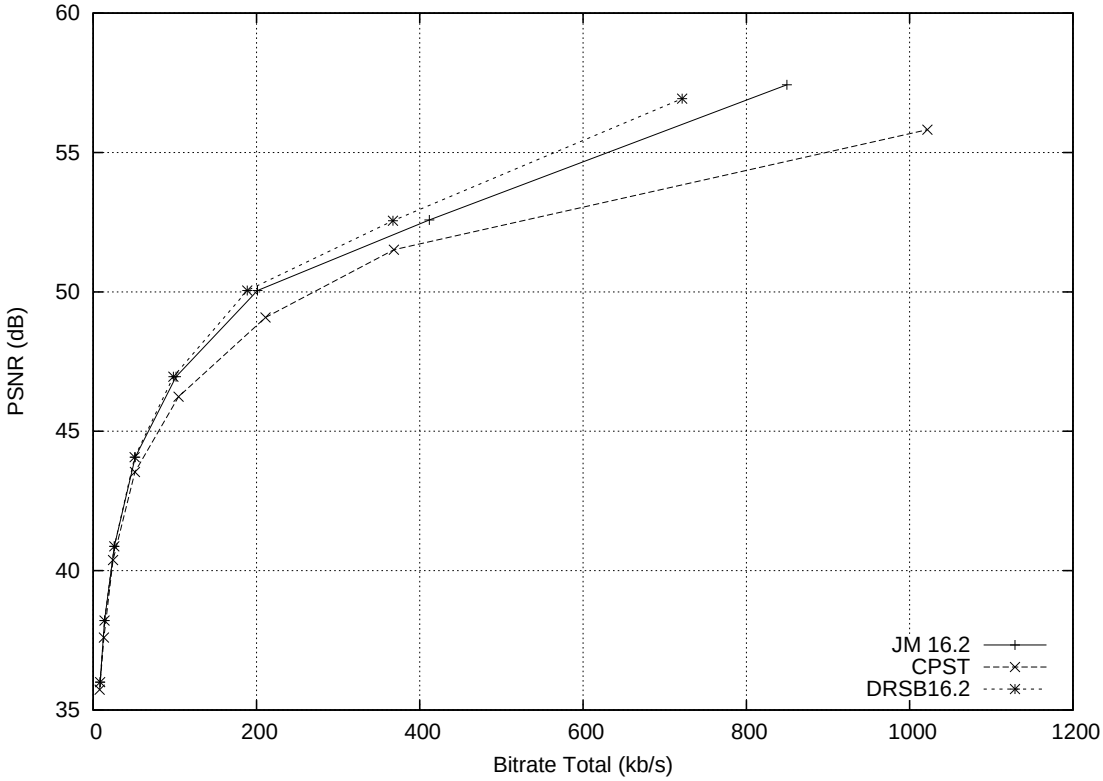


Figura A.49: *Akiyo* (QCIF) - Todos os quadros (Croma - U)

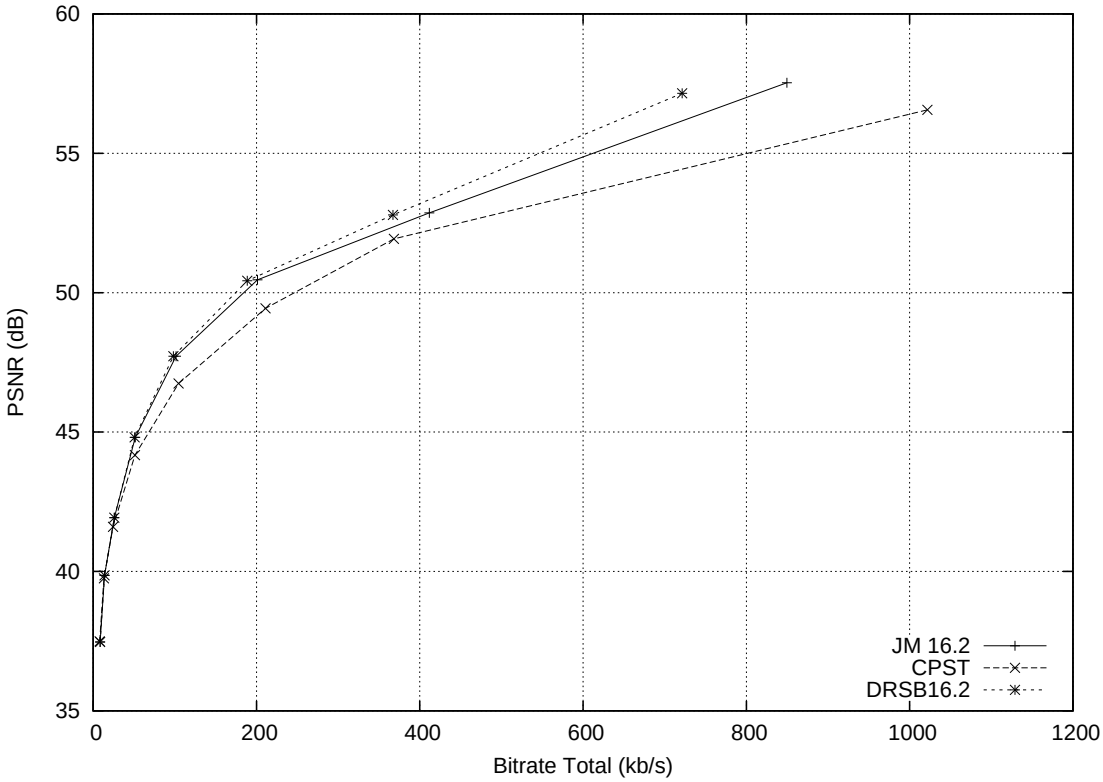


Figura A.50: *Akiyo* (QCIF) - Todos os quadros (Croma - V)

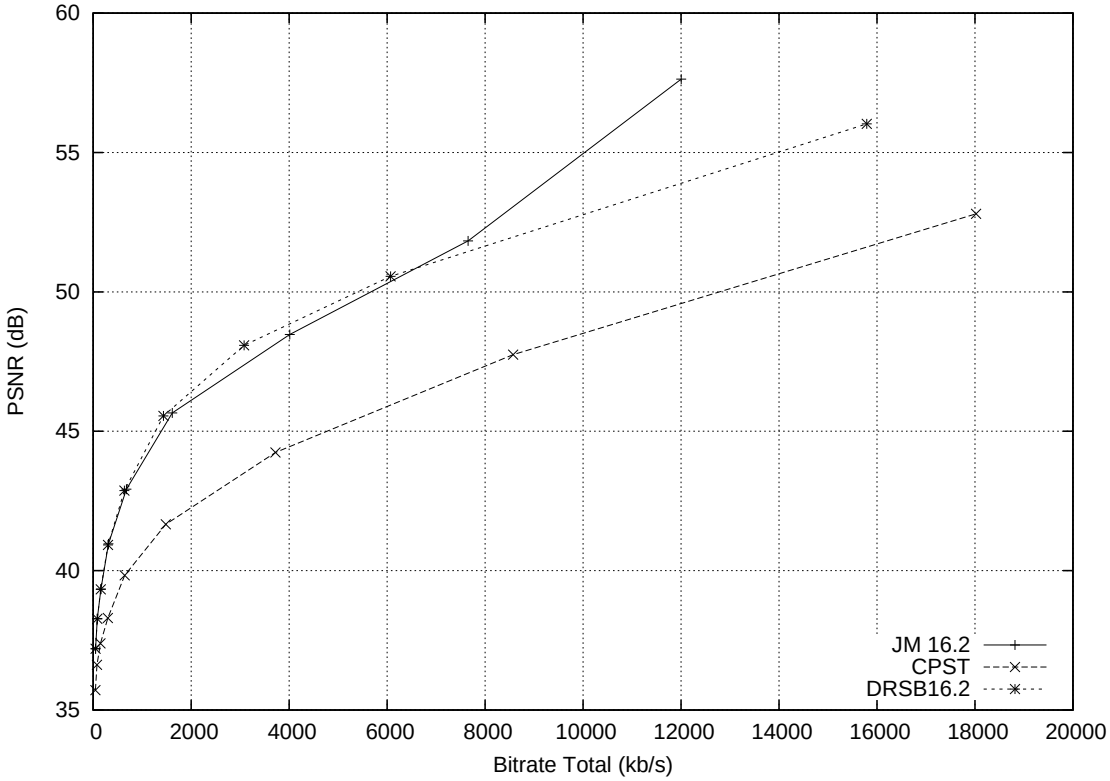


Figura A.51: *Foreman* (CIF) - Todos os quadros (Croma - U)

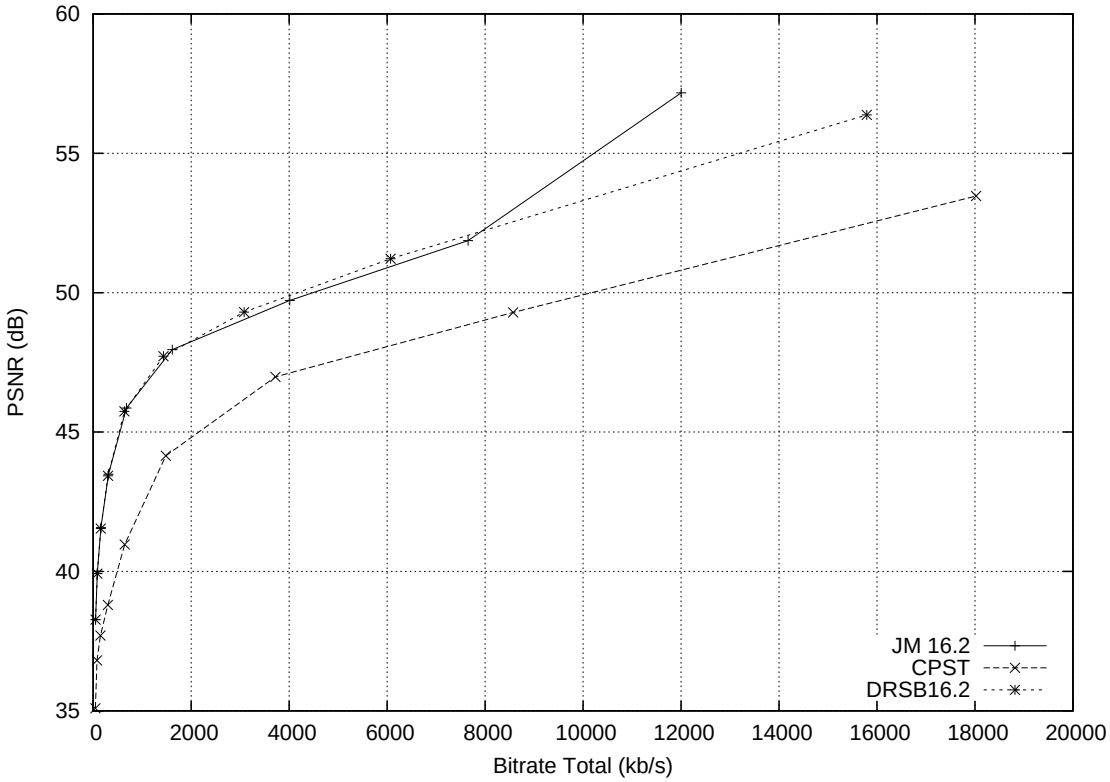


Figura A.52: *Foreman* (CIF) - Todos os quadros (Croma - V)

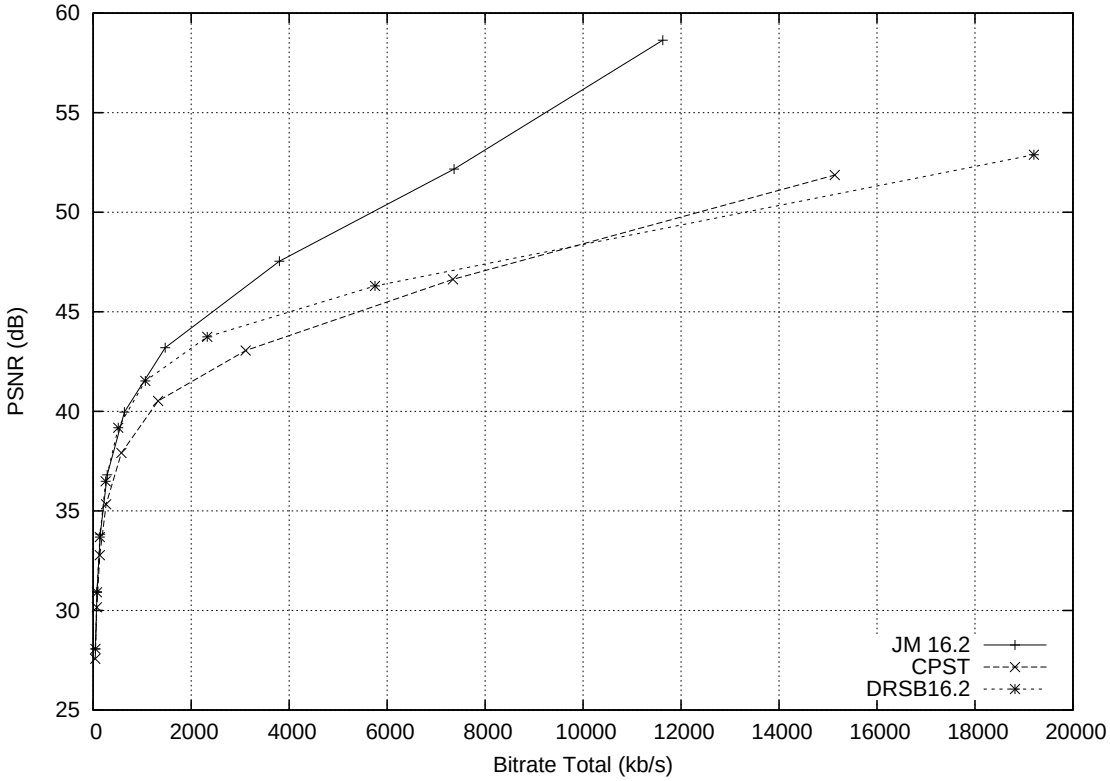


Figura A.53: *Foreman* (CIF) - Todos os quadros com GOP IBBBP (Luma)

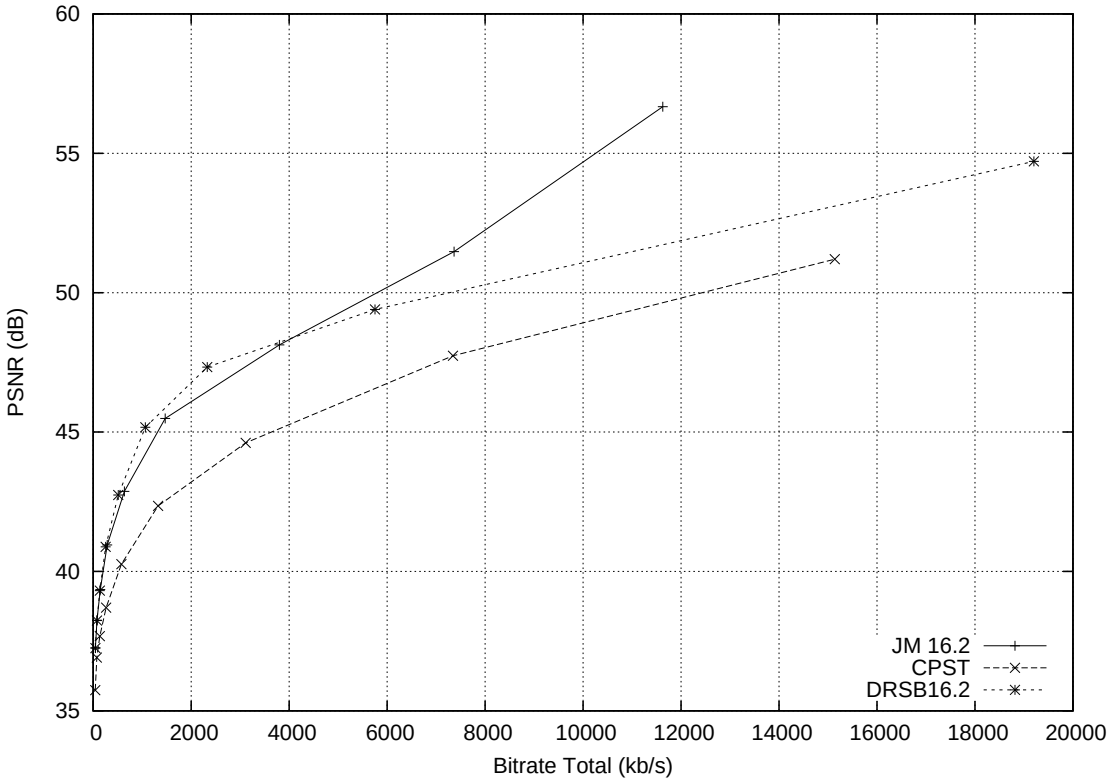


Figura A.54: *Foreman* (CIF) - Todos os quadros com GOP IBBBP (Croma - U)

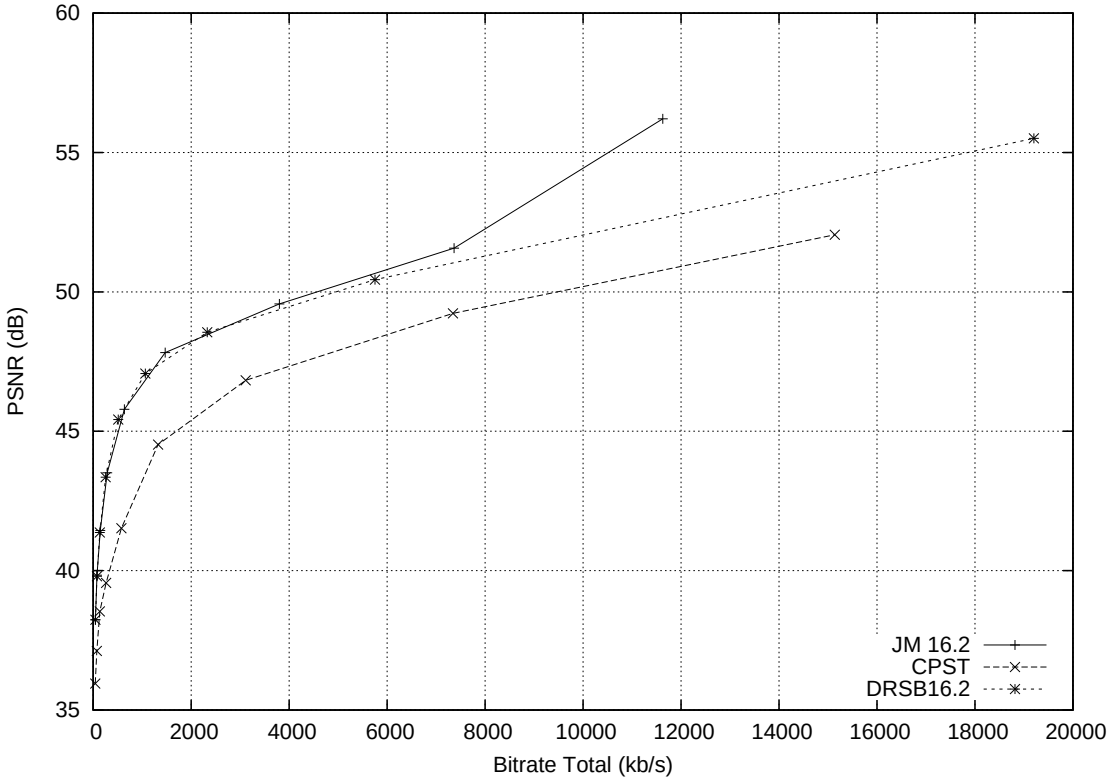


Figura A.55: *Foreman* (CIF) - Todos os quadros com GOP IBBBP (Croma - V)

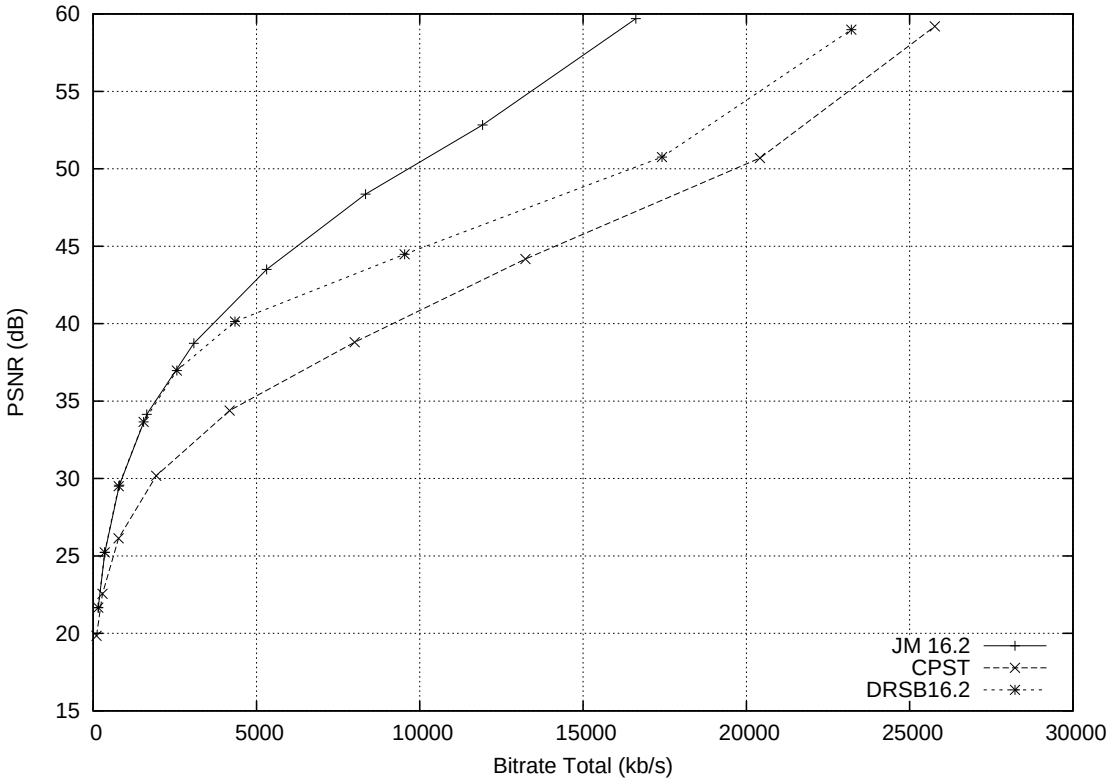


Figura A.56: *Flower Garden* (SIF) - Todos os quadros (Luma)

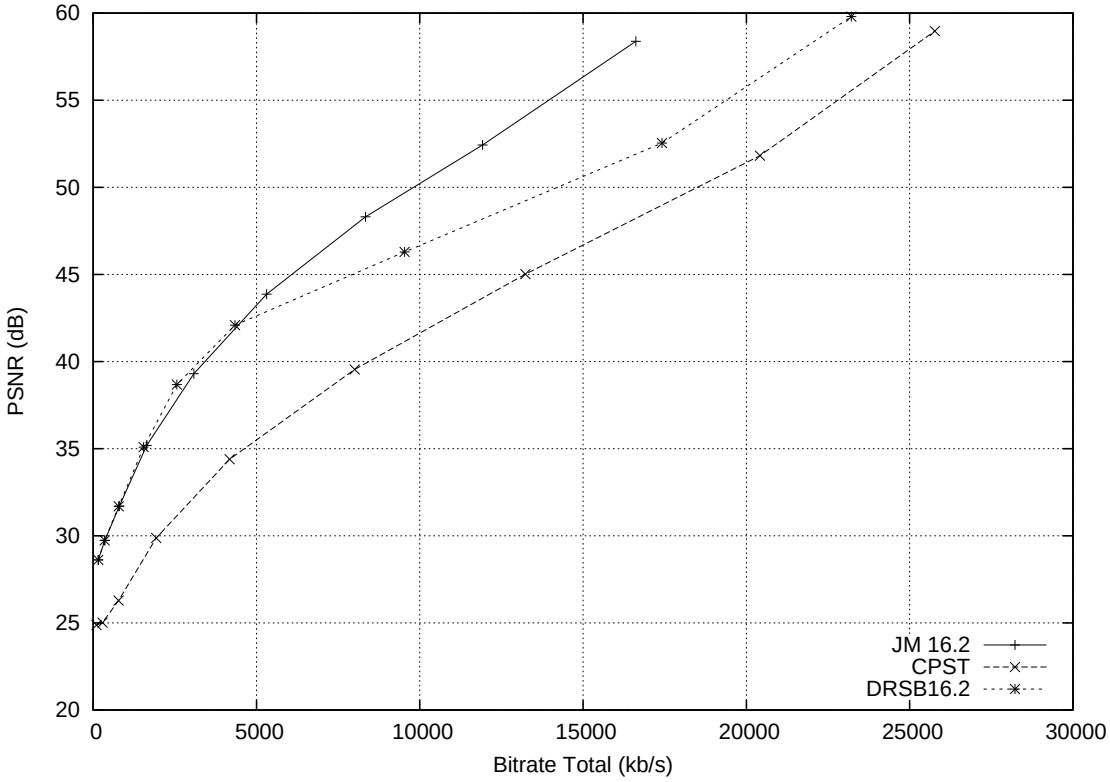


Figura A.57: *Flower Garden* (SIF) - Todos os quadros (Croma - U)

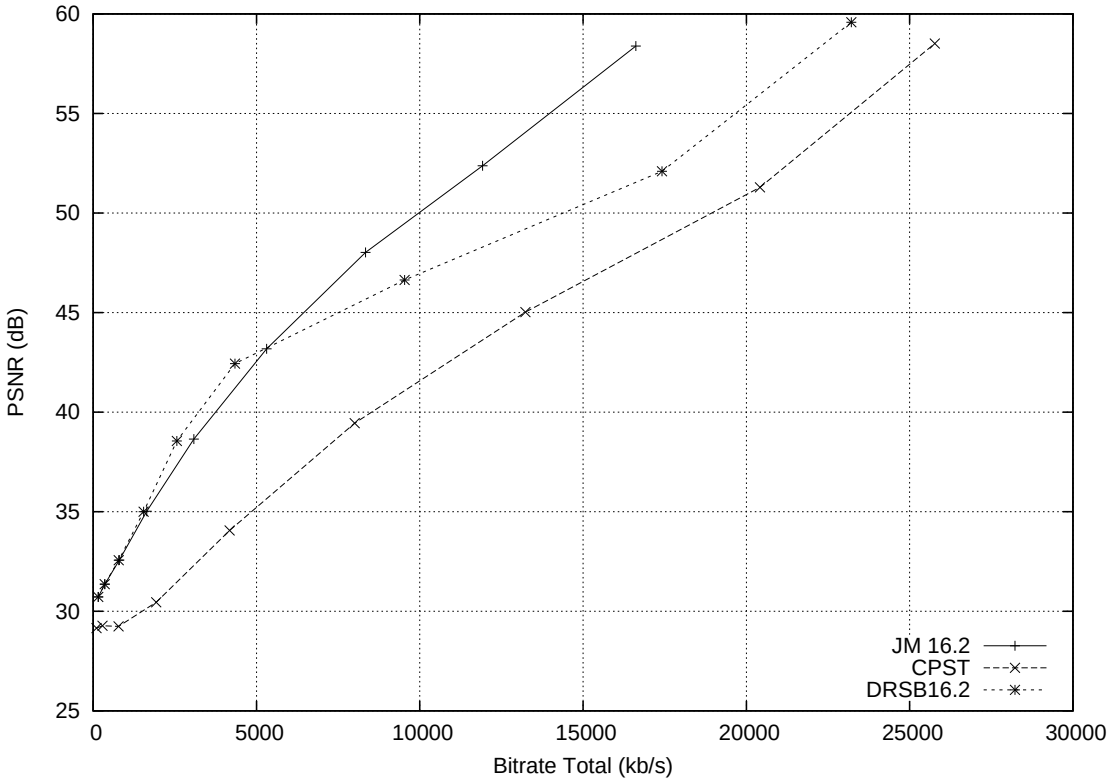


Figura A.58: *Flower Garden* (SIF) - Todos os quadros (Croma - V)

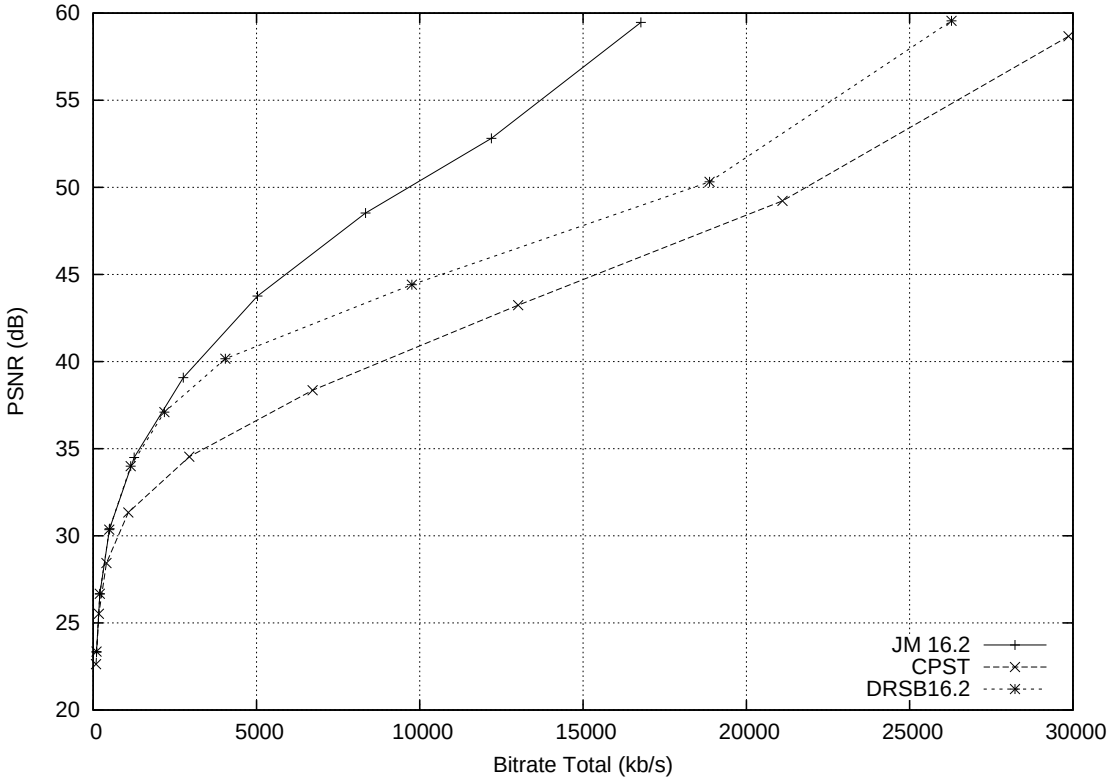


Figura A.59: *Mobile & Calendar* (CIF) - Todos os quadros (Luma)

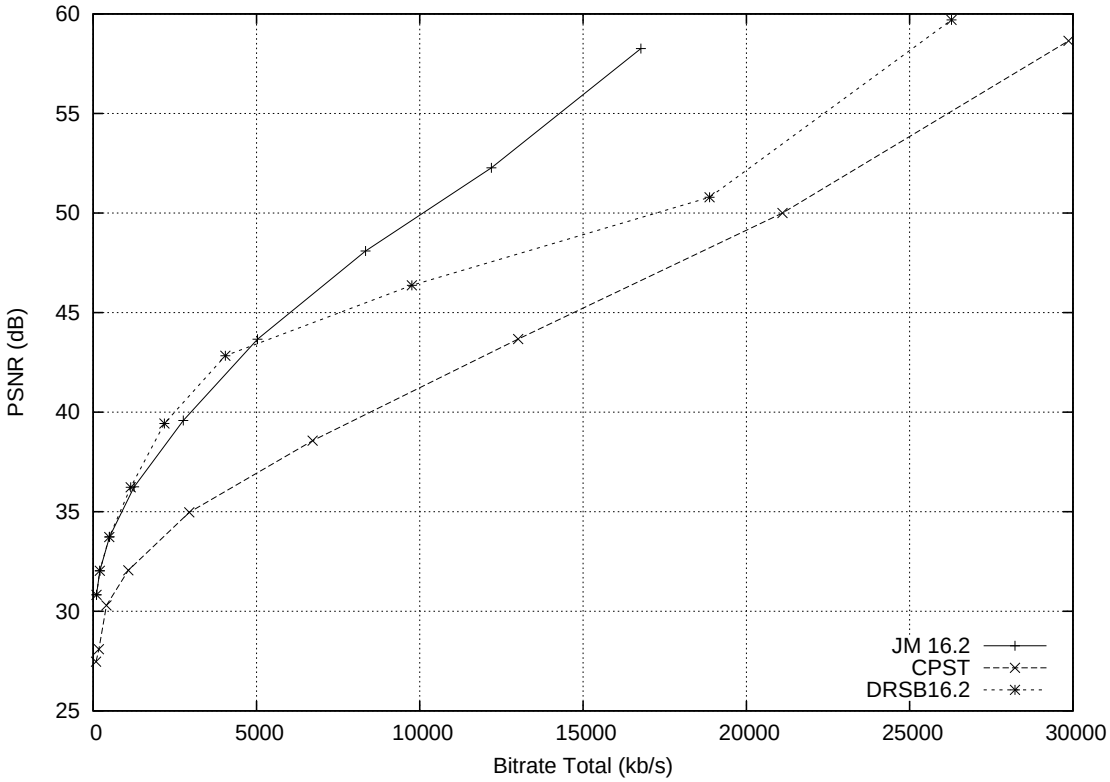


Figura A.60: *Mobile & Calendar* (CIF) - Todos os quadros (Croma - U)

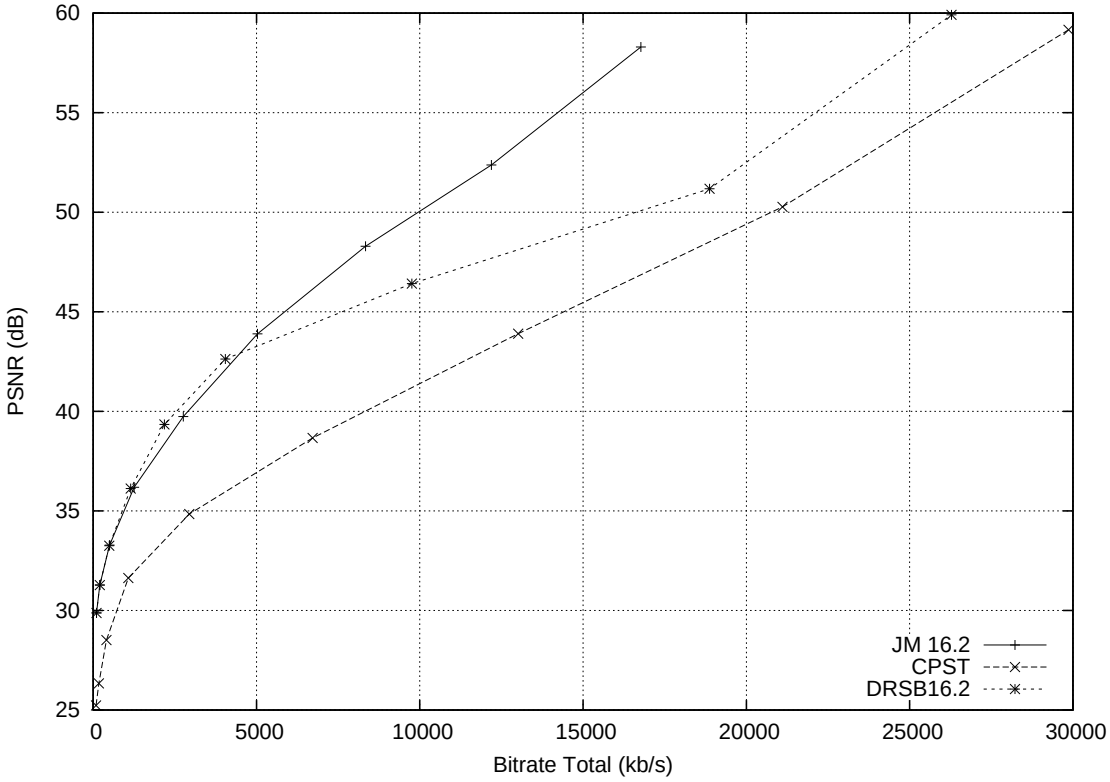


Figura A.61: *Mobile & Calendar* (CIF) - Todos os quadros (Croma - V)

A.3.2 Somentes Quadros com *slices* do Tipo P

Gráficos com informações de todos os quadros contendo *slices* do tipo P de uma mesma sequência para os métodos CPST e DRSB16.2 para as componentes de croma.

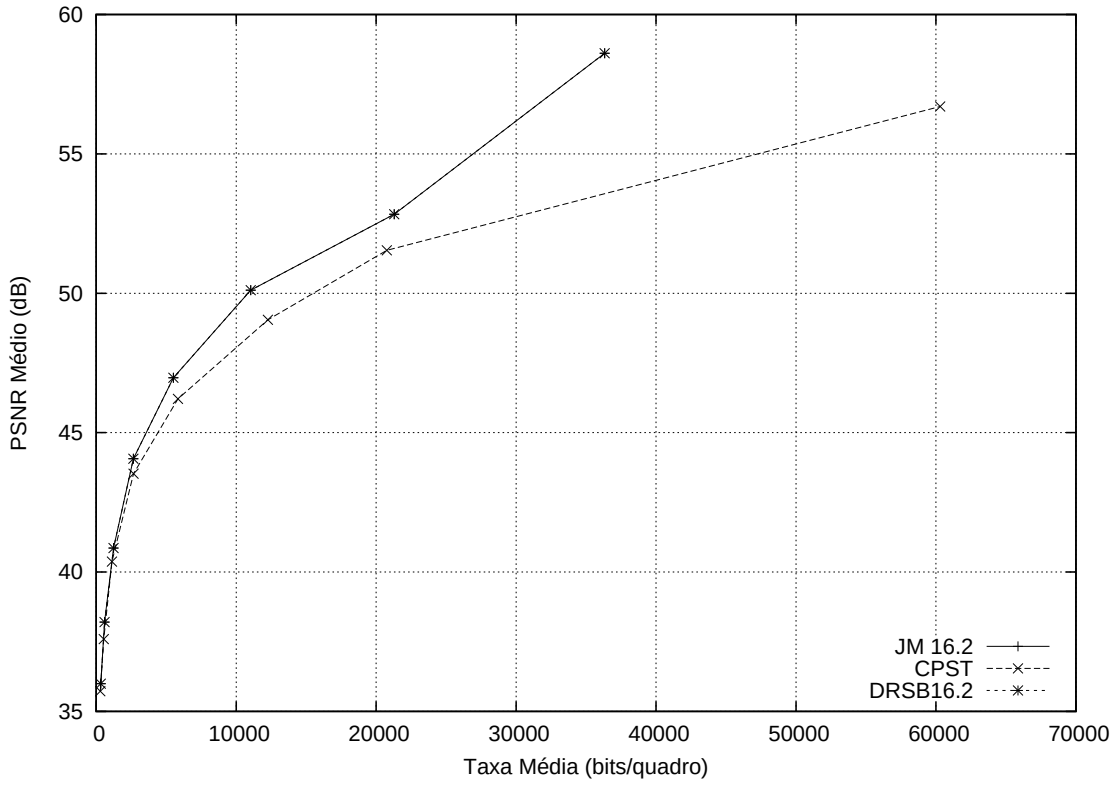


Figura A.62: *Akiyo* (QCIF) - Somente quadros com slices do tipo P (Croma - U)

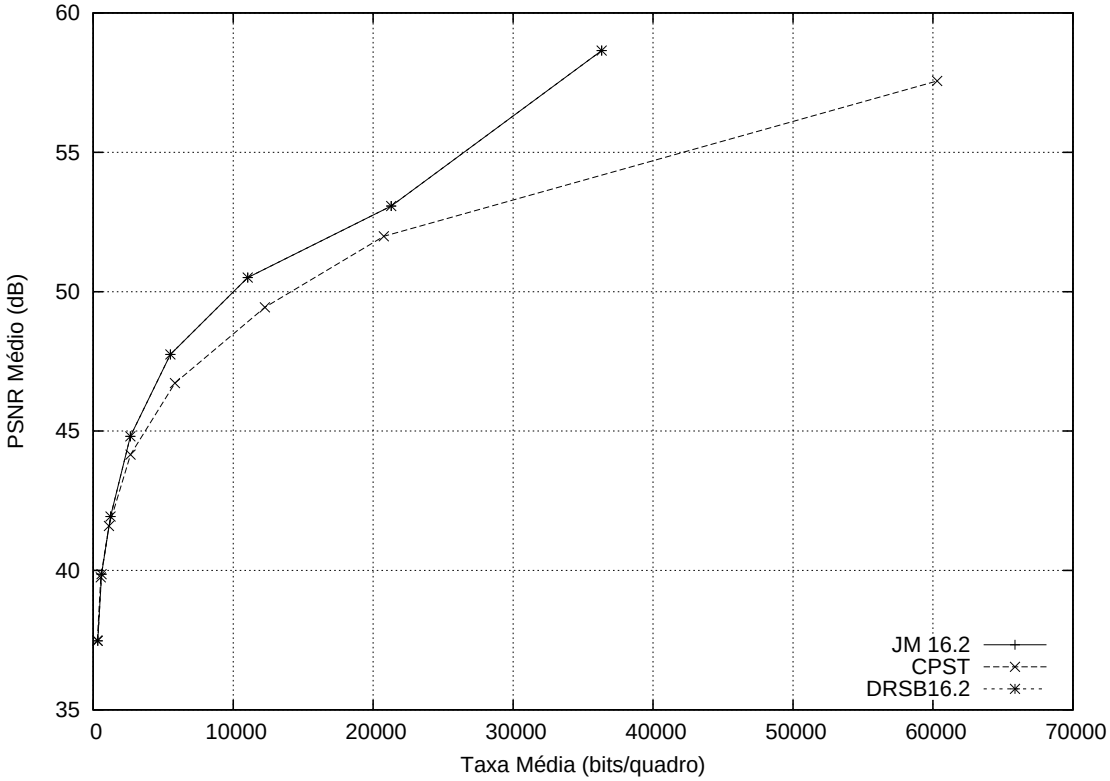


Figura A.63: *Akiyo* (QCIF) - Somente quadros com slices do tipo P (Croma - V)

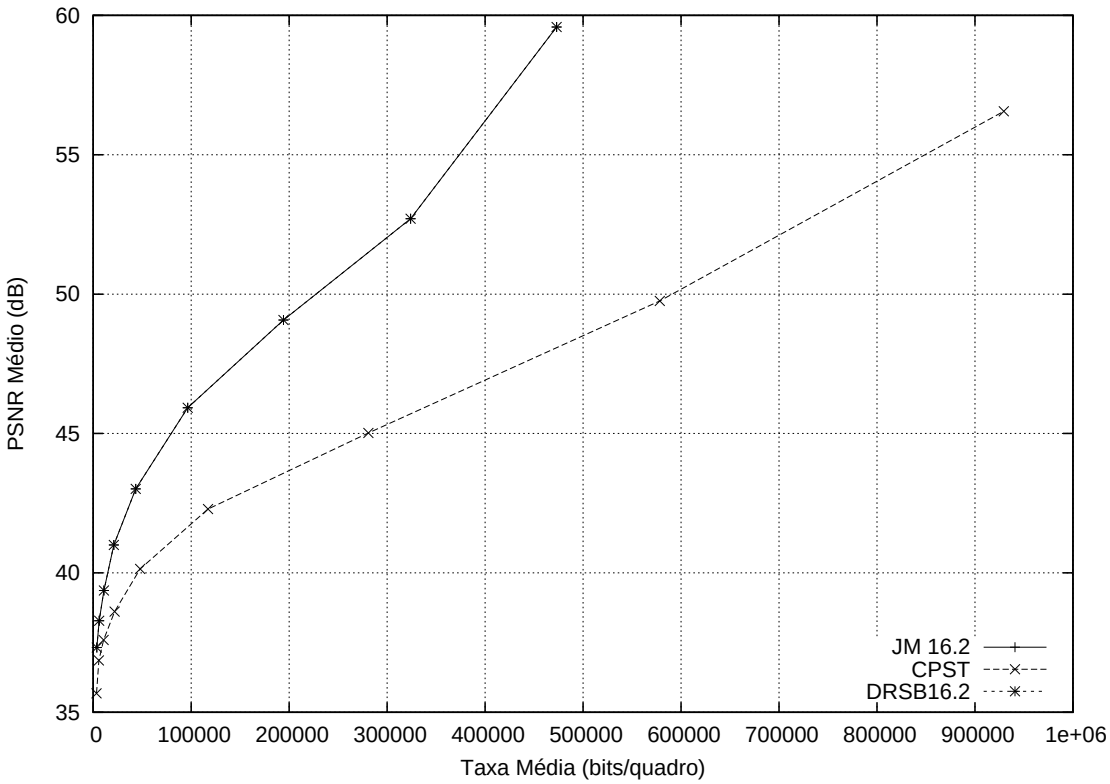


Figura A.64: *Foreman* (CIF) - Somente quadros com slices do tipo P com GOP IBBBP (Croma - U)

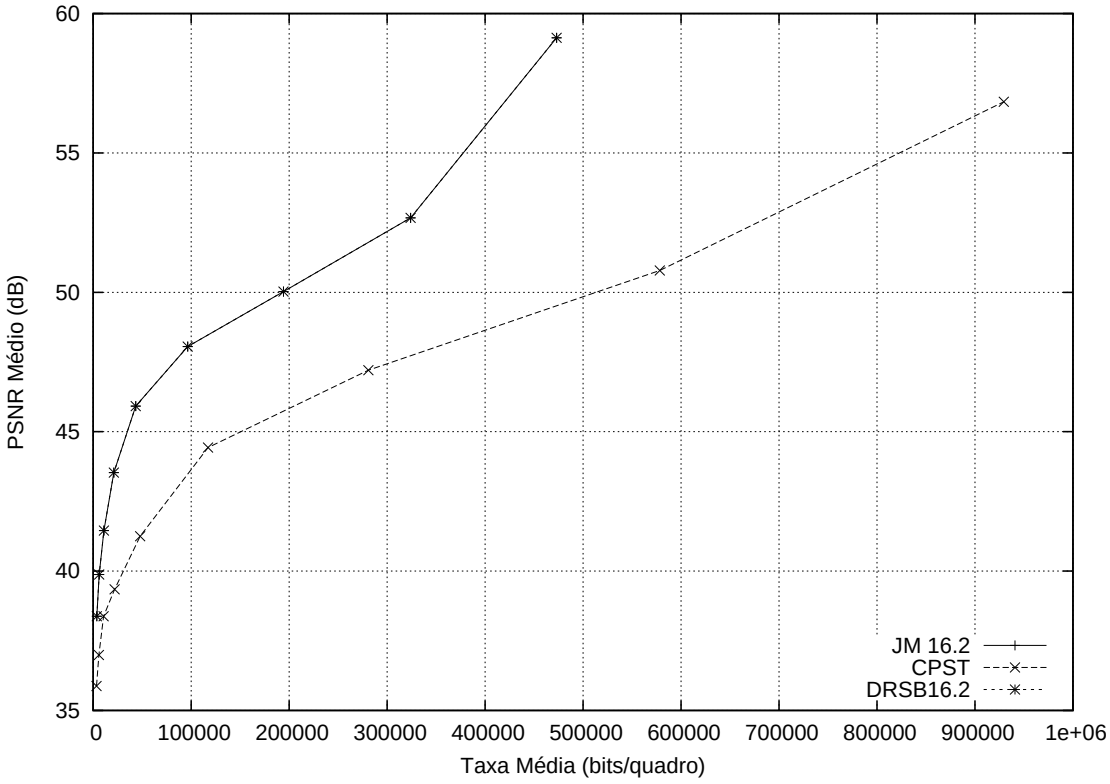


Figura A.65: *Foreman* (CIF) - Somente quadros com slices do tipo P com GOP IBBBP (Croma - V)

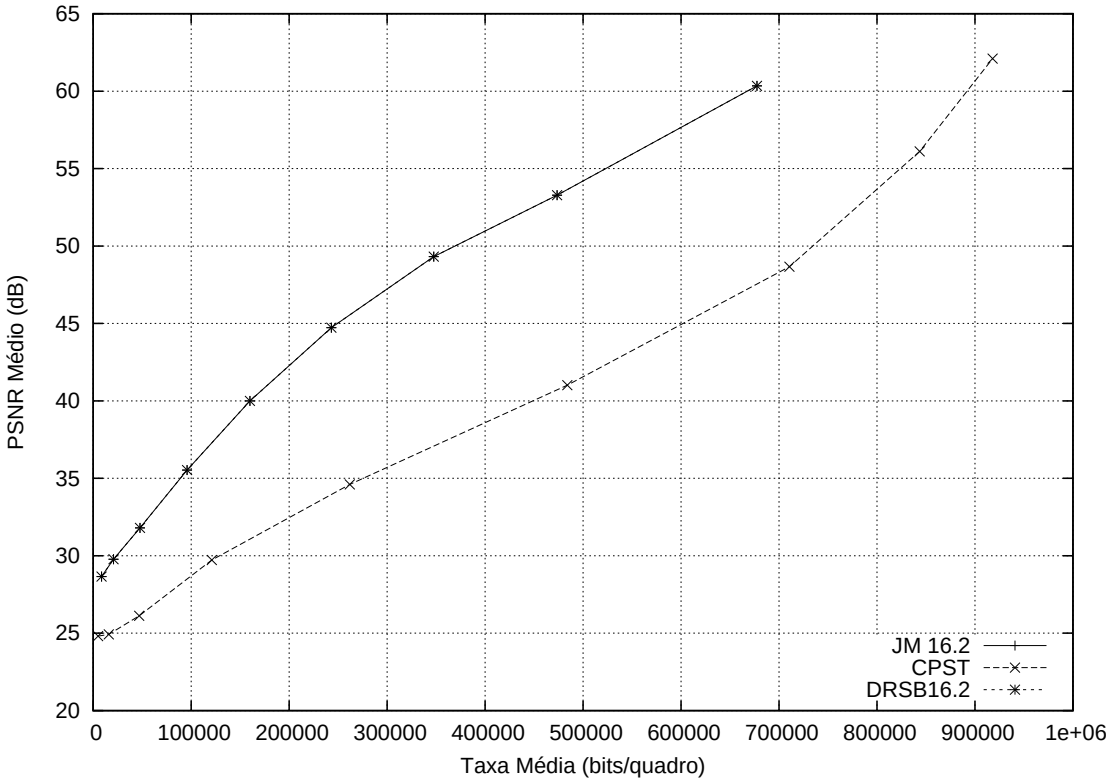


Figura A.66: *Flower Garden* (SIF) - Somente quadros com slices do tipo P (Croma - U)

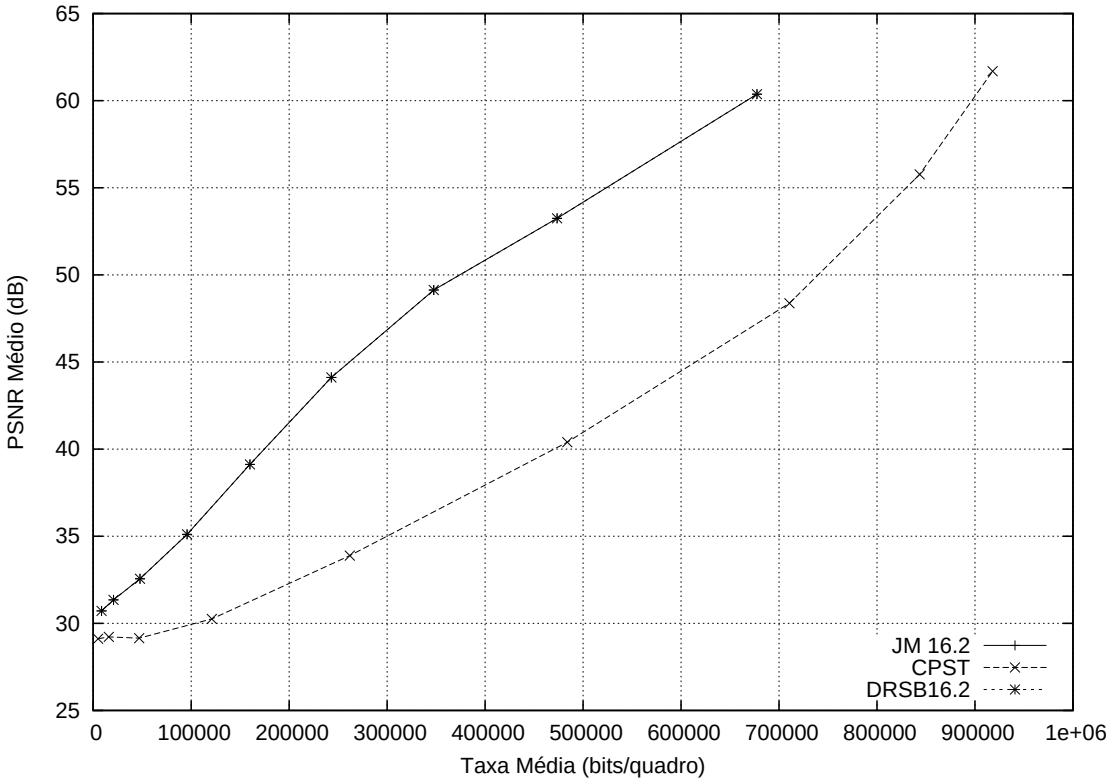


Figura A.67: *Flower Garden* (SIF) - Somente quadros com slices do tipo P (Croma - V)

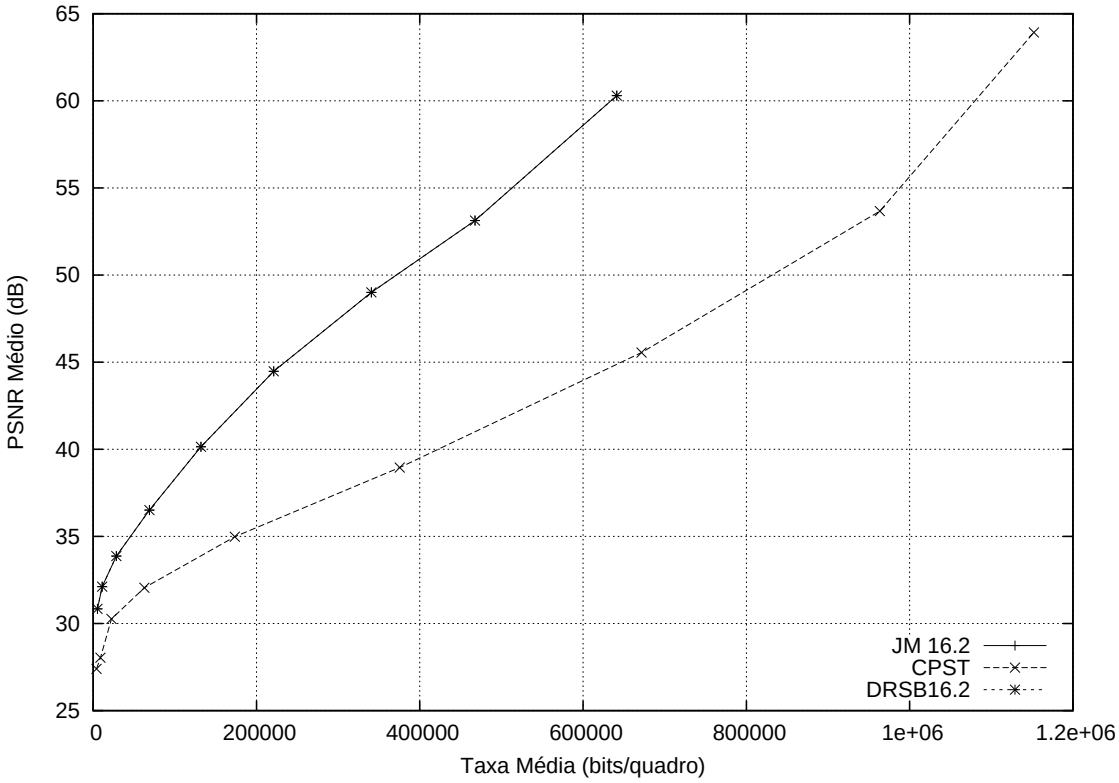


Figura A.68: *Mobile & Calendar* (CIF) - Somente quadros com slices do tipo P (Croma - U)

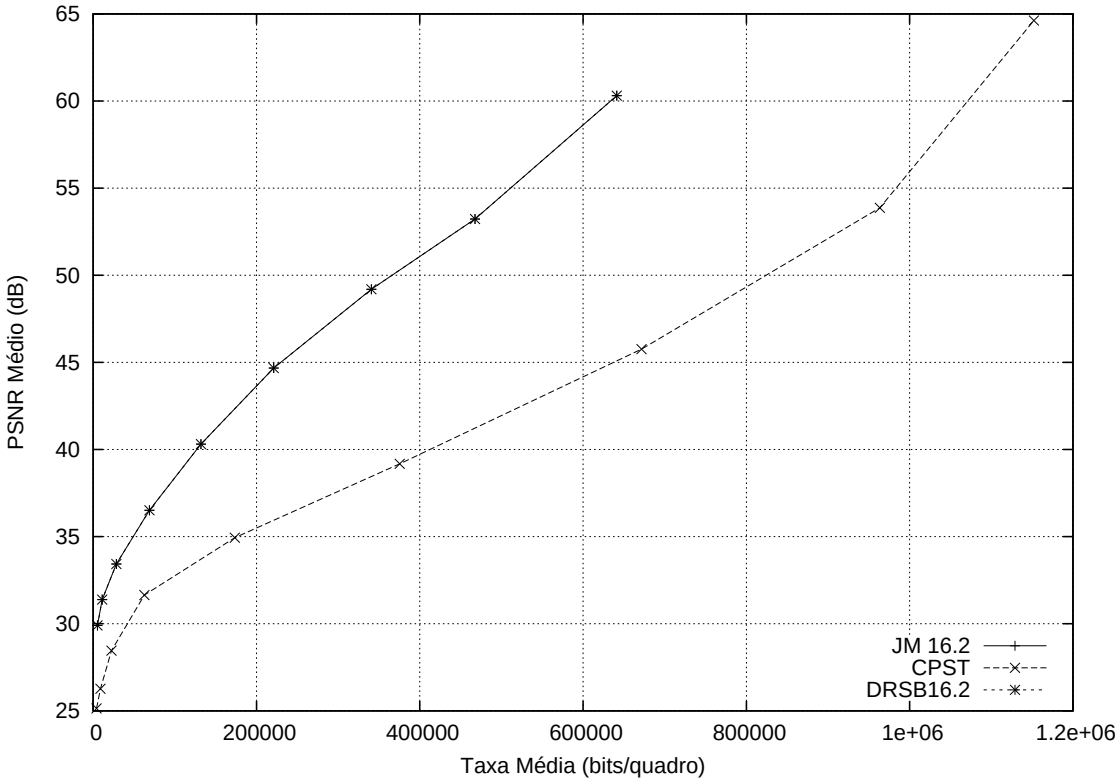


Figura A.69: *Mobile & Calendar* (CIF) - Somente quadros com slices do tipo P (Croma - V)

A.3.3 Somentes Quadros com *slices* do Tipo B

Gráficos com informações de todos os quadros contendo *slices* do tipo B de uma mesma sequência para os métodos CPST e DRSB16.2 para as componentes de croma.

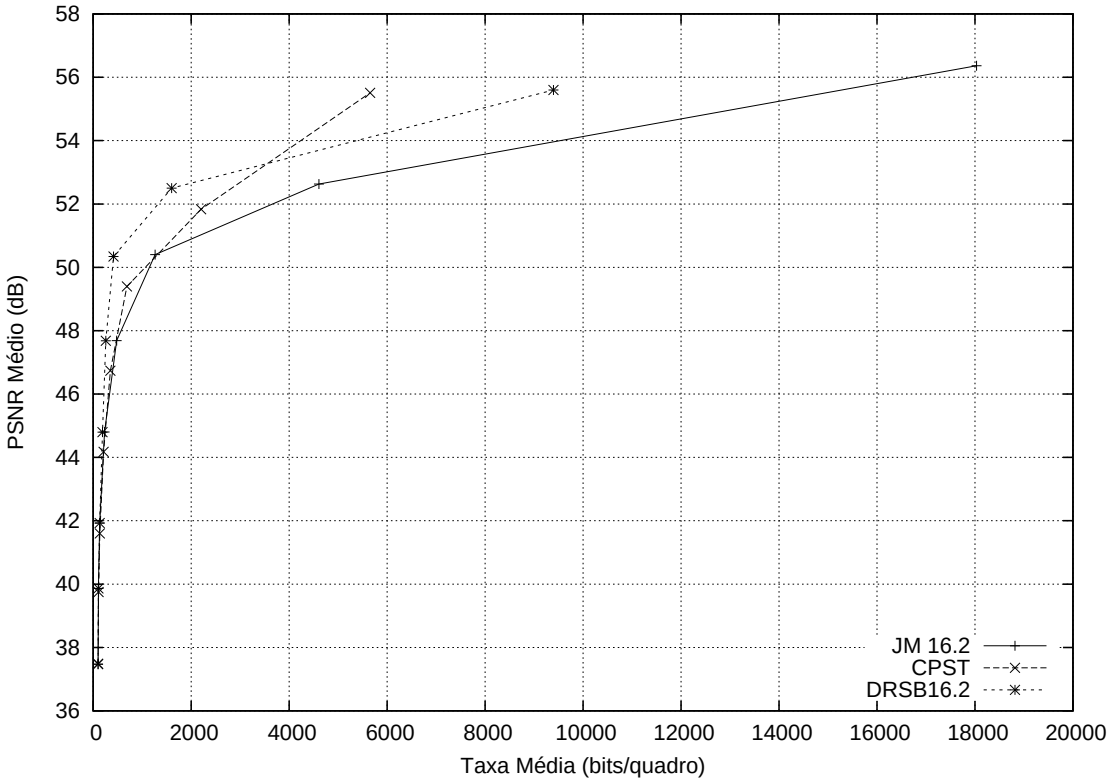


Figura A.71: *Akiyo* (QCIF) - Somente quadros com slices do tipo B (Croma - V)

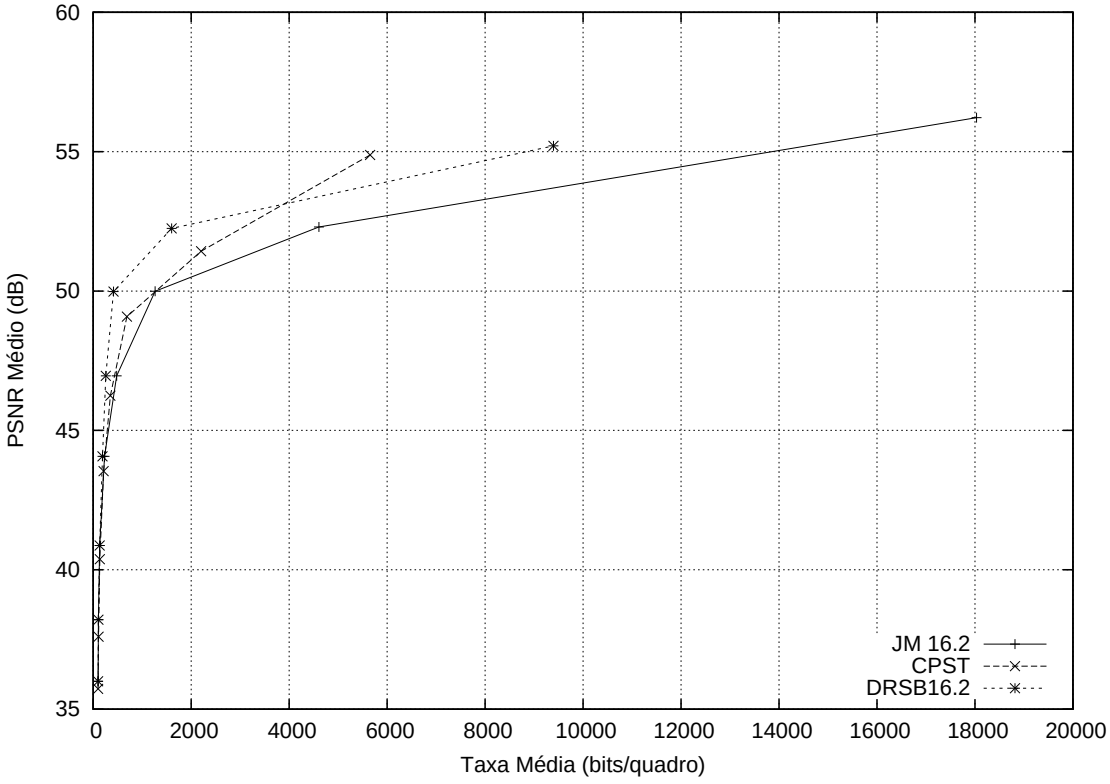


Figura A.70: *Akiyo* (QCIF) - Somente quadros com slices do tipo B (Croma - U)

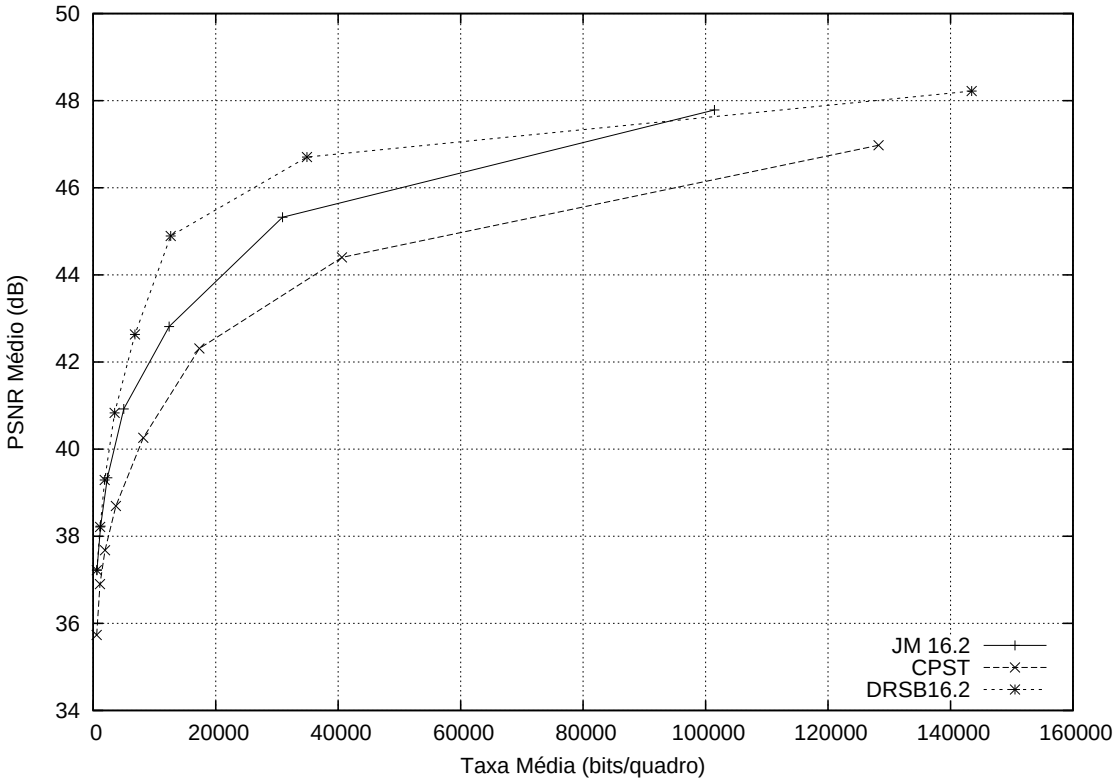


Figura A.72: *Foreman* (CIF) - Somente quadros com slices do tipo B com GOP IBBBP (Croma - U)

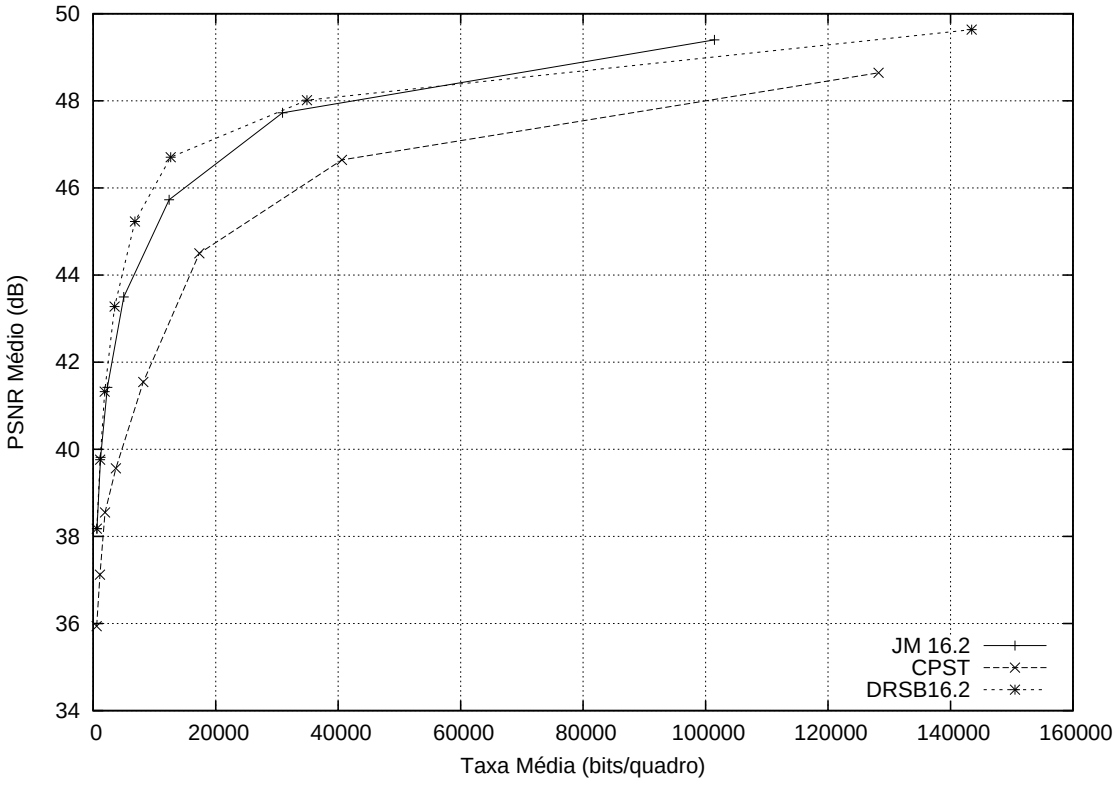


Figura A.73: *Foreman* (CIF) - Somente quadros com slices do tipo B com GOP IBBBP (Croma - V)

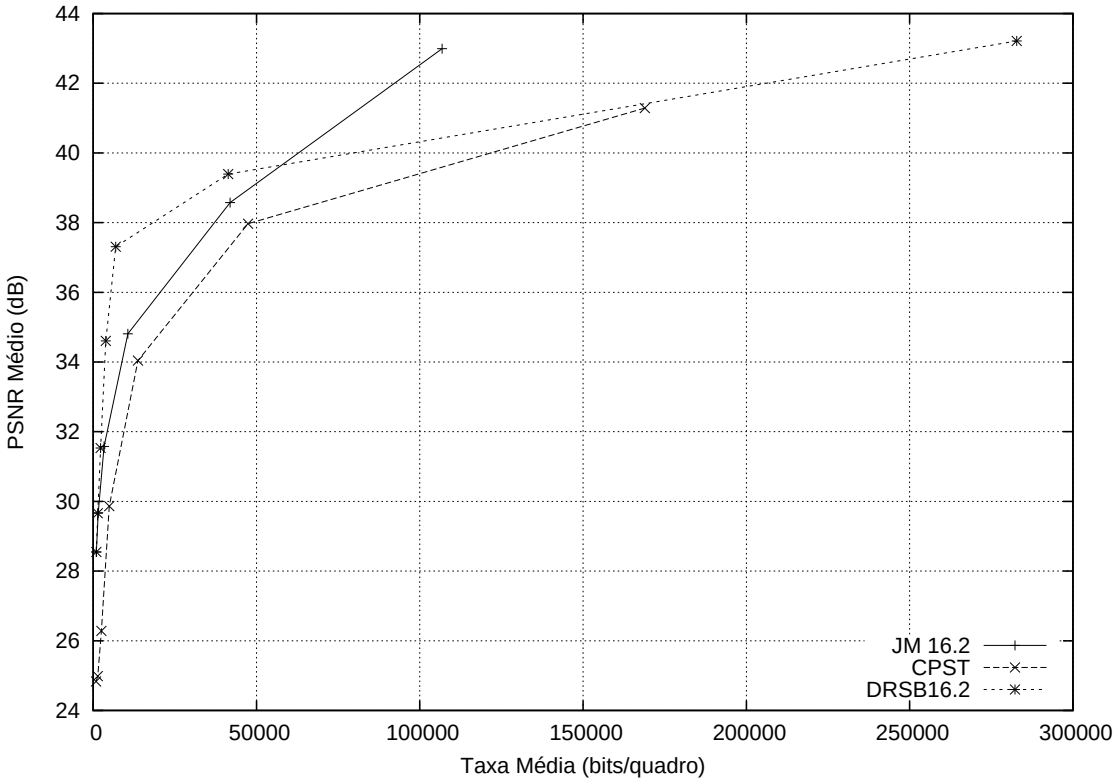


Figura A.74: *Flower Garden* (SIF) - Somente quadros com slices do tipo B (Croma - U)

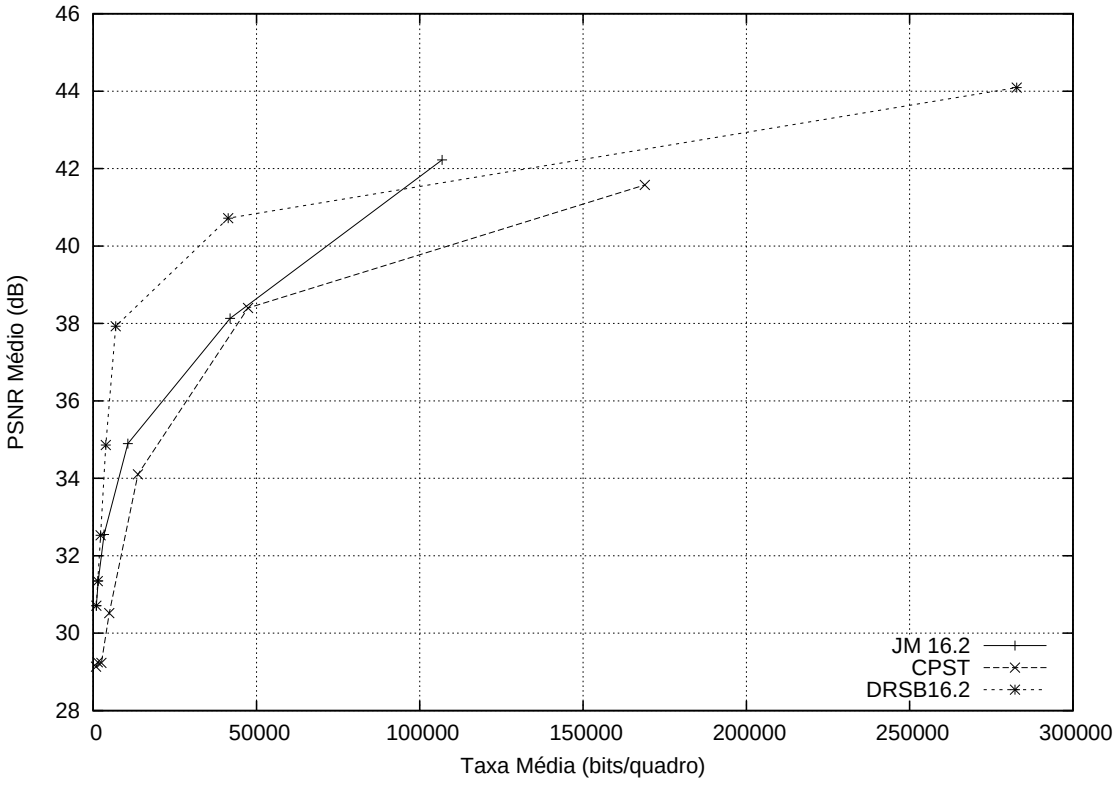


Figura A.75: *Flower Garden* (SIF) - Somente quadros com slices do tipo B (Croma - V)

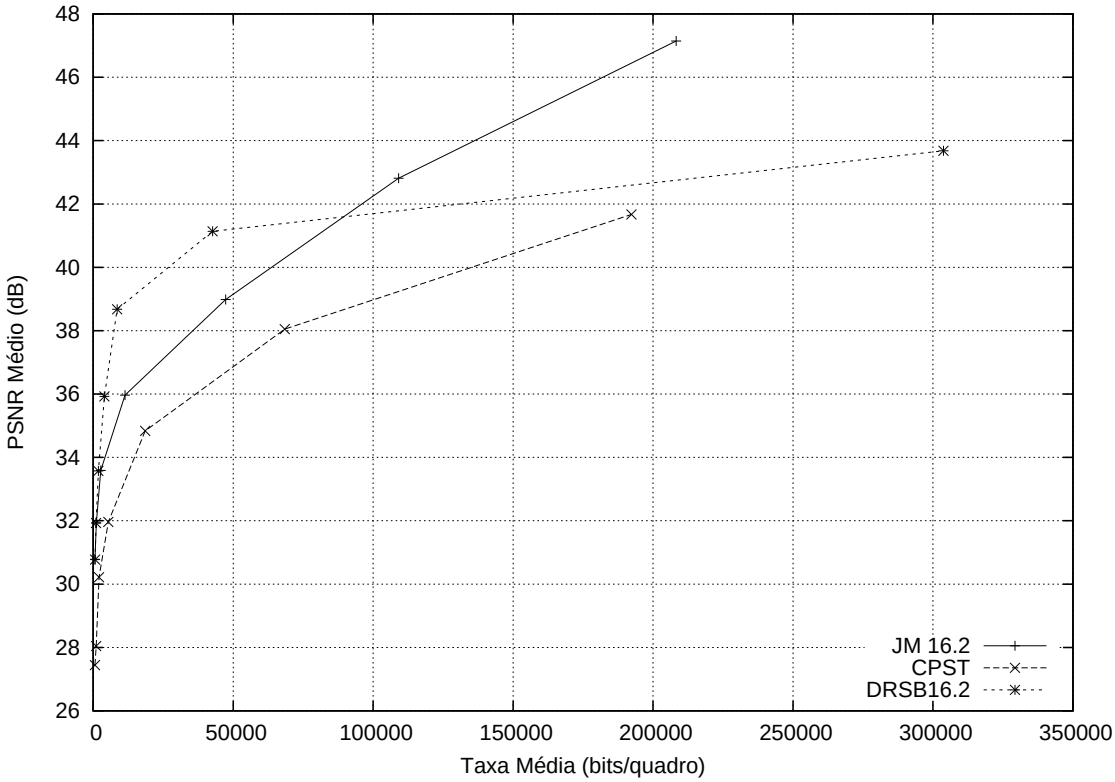


Figura A.76: *Mobile & Calendar* (CIF) - Somente quadros com slices do tipo B (Croma - U)

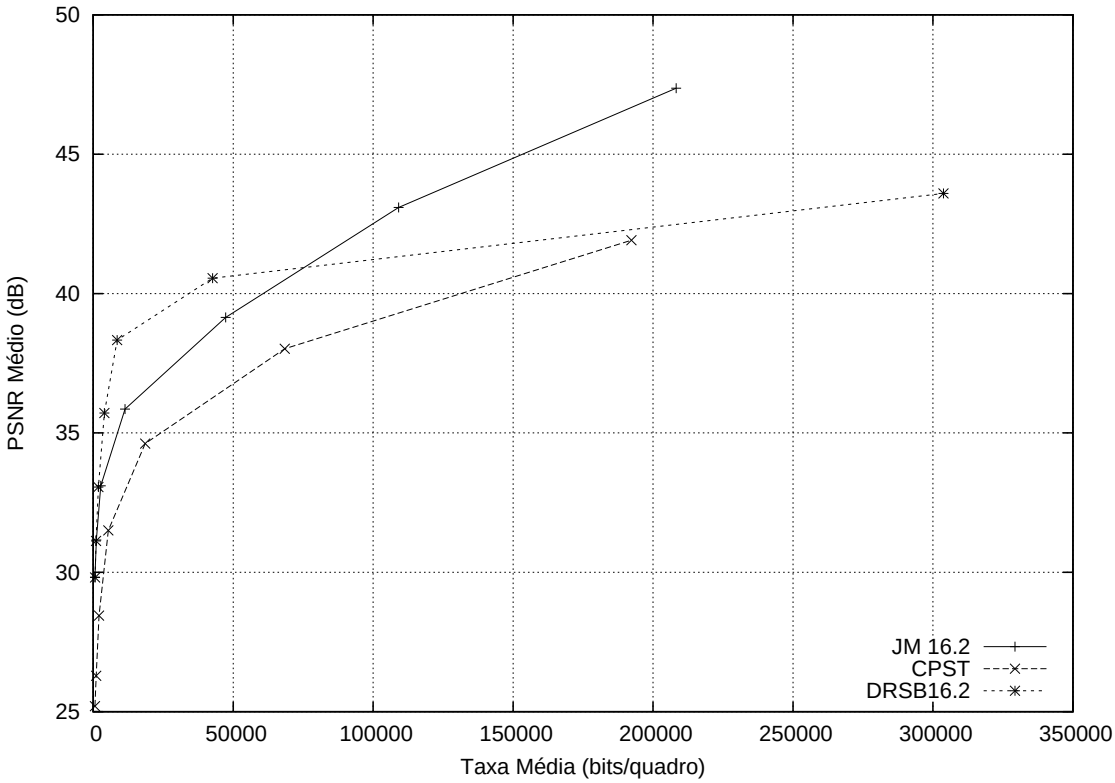


Figura A.77: *Mobile & Calendar* (CIF) - Somente quadros com slices do tipo B (Croma - V)