Resource allocation for H.264 video encoder:
Sequence-level controller

Alexander Bronstein*

September 7, 2009

Abstract

Tradeoff between different resources (bitrate, computational time, etc.) and compression quality, often referred to as rate-distortion optimization (RDO), is a key problem in video coding. With a little exaggeration, it can be claimed that what distinguishes between a good and a bad codec is how optimally it finds such a tradeoff. This is the first part of our paper presenting an algorithm for resource allocation in a scalable H.264 video codec. Here, we discuss a sequence-level resource allocation and control scheme. The scheme allows for optimal frame rate and frame dropping decision, optimal reference frame selection, adaptation to a wide range of bitrates and computational load balancing between different frames.

1 Introduction

The goal of video compression, in simple words, is to deliver a video of as high quality as possible using as few bits and utilizing as few computations as possible. Since these desiderata are conflicting, tradeoff between different resources and compression quality is a key problem in video coding. Usually referred to as rate-distortion optimization (RDO), the problem is actually broader and in addition to bitrate can include additional resources such as computational time or power dissipation [2, 1]. With a little exaggeration, it can be said that what distinguished a good video codec from a bad one is the mechanism performing such a tradeoff.

The target application we consider as a motivation for this paper is video distribution in a modern networked ecosystem of consumer electronic devices. A typical high-definition (HD) video arrives to our home from a cable or satellite broadcast, usually compressed using the MPEG-2 standard, at a bitrate up

*Department of Computer Science, Technion – Israel Institute of Technology, Haifa, 32000, Israel.
to 20 Mbps. After transcoding, this video source can then be distributed to multiple destinations: over the local wireless network to a local networked or portable device, or over the broadband to a remote Internet-connected location. The target bitrates in these two cases can vary dramatically: the bandwidth of a wireless network can typically range from 10 to 2 Mbps (whether using 801.11n or 801.11g standards), while the typical bandwidth of a broadband connection is 200 – 500 Kbps. To make the problem even more complicated, the available resource may change dynamically: the bandwidth may drop depending on the network utilization by other users and applications, and the available computational resources may vary depending, for example, of the number of channels simultaneously encoded.

The described video distribution scenario imposes two important requirement on the encoder. First, there are multiple operating points in the resource space (corresponding to different bandwidth and computational complexity). The encoder must be able to work at any of these points, or in other words, be scalable. Second, since the resources may change in time, the encoder must be able to transition from one operating point to another as quickly as possible. Finally, given the available resources, the quality of the video should be as high as possible. All of the above make the resource allocation and control a challenging and non-trivial problem.

Here, we discuss a sequence-level resource allocation and control scheme. As a reference, we use the H.264 standard, though the concepts are generic and can be applied to other MPEG-type encoding standards.

This paper is organized as follows. Section 2 presents a general background on video compression and the problem of resource allocation and control. In Section 3, we present a system-level view and the main features of the proposed resource allocation and control scheme, focusing on the sequence-level controller. Section 5 describes the buffer models. Section 4 describes the models of PSNR, bit production, encoding complexity, as well as their training and adaptation mechanisms. In Section 6, we describe the flow of the sequence-level control algorithm. Section 7 deals with sequence resource allocation. Sections 9 and 10 are dedicated to frame encoding and encoding order optimization.

2 Background

2.1 General principles of video coding

Video compression takes advantage of these redundancies, trying to reduce the size of the video data while maintaining the content almost without visible loss. Such compression is referred to as lossy (as opposed to lossless), since the original data cannot be recovered exactly. Most modern codecs operate by constructing
a prediction of the video frame by using the redundant information. The prediction parameters are usually much more compactly represented than the actual pixels. The predicted frame is usually similar to the original one, yet, differences may appear at some pixels. For this purpose, the residual (prediction error) is transmitted together with the prediction parameters. The error is also usually compactly represented using spatial transform, like the discrete cosine transform (DCT) with subsequent quantization of the transform coefficients. The quantized coefficients and the prediction parameters are entropy-encoded to reduce furthermore their redundancy.

Inter-prediction accounts for the major percentage of compression of the video data size in video codecs. It takes advantage of temporal redundancy, predicting a frame from nearby frames (referred to as reference frames) by moving parts of these frames, thus trying to compensate for the motion of the objects in the scene. The description of the motion (motion vectors) are the parameters of the prediction which are transmitted. This type of prediction is used in MPEG-like codecs. The process of producing intra-prediction is called motion estimation. Typically, it is performed in a block-wise manner, when many modern codec support motion estimation with blocks of adaptive size.

At the other end, intra-prediction takes advantage of spatial redundancy and predicts portions of a frame (blocks) from neighbor blocks within the same frame. Such prediction is usually aware of spatial structures that may occur in a frame, namely smooth regions and edges. In the latter case, the prediction is directional. Such type of prediction is used in the recent H.264 Advances Video Codec (AVC) standard [?] and does not appear in previous versions of the MPEG standard.

The actual reduction in the amount of transmitted information is performed in the transmission of the residual. The residual frame is divided into blocks, each of which undergoes a DCT-like transform. The transform coefficients undergo quantization, usually performed by scaling and rounding. Quantization allows represent the coefficients by using less precision, thus reducing the amount of information required. Quantized transform coefficients are transmitted using entropy coding. This type of coding a lossless and utilizes further redundancy that may exist in the transform coefficients. The encoder usually operates with video represented in the YCbCr color space, where the achromatic Y channel is called the luma and the chromatic channels (Cb and Cr) are called chroma. The chroma is typically downsampled to half frame size in each direction, which is referred to as the 4 : 2 : 0 format.

The performance of a lossy video codec is measured as the tradeoff between the amount of bits required to describe the data and the distortion introduced by the compression, referred to as the rate-distortion (RD) curve. The RD curve of a better codec is closer to the idealistic and theoretically impossible case of zero distortion and zero rate. As the distortion criterion, the mean squared error (MSE) of the luma channel is
usually used. The MSE is often converted into logarithmic units and represented as the *peak signal-to-noise ratio* (PSNR),

\[ p = 10 \log_{10}\left(\frac{y_{\text{max}}^2}{d}\right) \]  

(here \(d\) is the MSE and \(y_{\text{max}}\) is the maximum allowed value of the luma pixels, typically 255 if the luma data has an 8-bit precision and is represented in the range 0, ..., 255).

**2.2 H.264 video codec**

The H.264 AVC is one of the most recent standard in video compression, offering significantly better compression rate and quality compared to the previous MPEG-2 and MPEG-4 standards and targeted to high definition (HD) content. For example, H.264 delivers the same quality as MPEG-2 at a third to half the data rate.

The encoding process goes as follows: a frame undergoing encoding is divided into non-overlapping macroblocks, each containing 16 × 16 luma pixels and 8 × 8 chroma pixels (in the most widely used 4 : 2 : 0 format). Within each frame, macroblocks are arranged into slices, where a slice is a continuous raster scan of macroblocks. Each slice can be encoded independently of the other. Main slice types are P and I. An I slice may contain only I macroblocks; a P slice may contain P or I macroblocks. The macroblock type determines the way it is predicted. P refers to inter-predicted macroblocks; such macroblocks are subdivided into smaller blocks. I refers to intra-predicted macroblocks; such macroblocks are divided into 4 × 4 blocks (the luma component is divided into 16 blocks; each chroma component is divided into 4 blocks).

In I mode (intra-prediction), the prediction macroblock is formed from pixels in the neighbor blocks in the current slice that have been previously encoded, decoded and reconstructed, prior to applying the in-loop deblocking filter. The reconstructed macroblock is formed by imitating the decoder operation in the encoder loop. In P mode (inter-prediction), the prediction macroblock is formed by motion compensation from reference frames. The prediction macroblock is subtracted from the current macroblock. The error undergoes transform, quantization and entropy coding. According to the length of the entropy code, the best prediction mode is selected (i.e., the choice between an I or a P macroblock, the motion vectors in case of a P macroblock and the prediction mode in case of an I macroblock). The encoded residual for the macroblock in the selected best mode is sent to the bitstream.

A special type of frame referred to as *instantaneous data refresh* (IDR) is used as a synchronization
mechanism, in which the reference buffers are reset as if the decoder started “freshly” from the beginning of the sequence. IDR frame is always an I-frame. The use of IDR allows, for example, to start decoding a bitstream not necessarily from its beginning. A sequence of P and I frames between two IDRs is called a group of pictures (GOP). A GOP always starts with an IDR frame. The maximum GOP size is limited by the standard.

The operating point on the RD curve is controlled by the quantization parameter, determining the “aggressiveness” of the residual quantization and the resulting distortion. In the H.264 standard, the quantization parameter is an integer in the range 0, ..., 51, denoted here by $q'$. The quantization step doubles for every increment of 6 in $q'$. Sometimes, it is more convenient to use the quantization step rather than the quantization parameter, computed according to

$$q = 0.85 \cdot 2^{\frac{q'-12}{6}}. \quad (2)$$

In the following, we will use $q$ and $q'$ interchangeably.

### 2.3 Resource allocation and control

Since only the decoder is standardized in the MPEG standards, many aspects of the way in which the encoder produces a standard-compliant stream is left to the discretion of a specific encoder design. The main difference between existing encoders is the decision process carried out by the bitrate controller that produces the encoder control parameters. Usually, the encoder parameters are selected in a way to achieve the best tradeoff between video quality and bitrate of the produced stream. Controllers of this type are referred to as RDO.

Parameters controlled by the bitrate controller typically include: frame type and reference frame or frames if the frame is a P-frame on the sequence level, and macroblock type and quantization parameter for each macroblock on the frame level. For this reason, it is natural and common to distinguish between two levels of bitrate control: sequence- and frame-level. The sequence-level controller is usually responsible for the frame type selection and allocation of the bit budget for the frame, and the frame-level controller is responsible for selection of the quantization parameter for each macroblock within the frame.

In a broader perspective, RD optimization is a particular case of the optimal resource allocation problem, in which the available resources may be the bitrate, computational time, power dissipation, etc. are distributed in a way that maximizes some quality criterion. Hence, hereinafter we use the term resource allocation referring to RD optimization-type problems discussed in this paper. The resources we consider
specifically are computational time and bitrate.

Theoretically, optimal resource allocation requires running the encoding with different sets of parameters and selecting the best outcome. Such an approach is impossible due to a very large number of possible combinations of parameters, which result in a prohibitive computational complexity. Suboptimal resource allocation approaches usually try to model some typical behavior of the encoder as function of the parameters; if the model has an analytical expression which can be efficiently computed, the optimization problem becomes practically solvable. However, since the model is only an approximate behavior of the encoder, the parameters selected using it may be suboptimal.

3 System-level view

The presented resource allocation and control mechanism are generic and in principle can be used with any MPEG-type encoder. To be more specific, in the following we assume that the underlying encoding pipeline implements the H.264 AVC encoder, with independent slice encoders working simultaneously. Such an implementation is typical of real-time hardware encoders.

A system-level block diagram of the proposed controllers is shown in Figure 1. The system consists of the sequence-level controller, which is responsible for frame type and reference selection and allocation of the bit budget and initial quantization parameter for the currently encoded frame. The frame-level sequence controller allocates the resources on sub-frame level, in order to utilize the total frame bit budget and achieve the highest perceptual visual quality. For this purpose, a perceptual quality model and a PSNR shaping mechanism are employed. Both controllers are based on the encoder model, which predicts the behavior of the encoder given a set of encoding parameters. The model, in turn, utilizes visual clues, a set of descriptors of the video data.

The main focus of this paper are the encoder models and the sequence-level controller. Among the most important features of the proposed sequence-level controller are:

**Content-aware encoder model**: a accurate encoder model based on visual clues, allowing to optimize the resource allocation prior to encoding;

**Optimal frame type decision**, including IDR insertion at scene cuts; optimization of frame dropping capable of detecting and removing redundant frames e.g. due to telecine conversion; and optimal reference frame selection based on frame predictability, allowing to achieve compression ratios close to those of codecs using multiple reference frames at significantly lower computational complexity;
Optimal encoding order decision, allowing out-of-order encoding to achieve better predictability.

Optimal frame bit allocation

Optimal encoding time allocation, allowing to achieve better encoding quality by distributing encoding time between frames according to their coding complexity.

4 Encoder model

The purpose of encoder model is to estimate the amount of resources used by the encoder without resorting to the expensive process of encoding itself. Specifically, we predict the amount of bits produced by the encoder, the distortion as the result of encoding and the time complexity of the encoding process.

First, the model gives explicit expressions to the predicted values (amount of bits, distortion, time) as function of encoder parameters ($q'$ or $q$, frame type, etc). Second, the model is adaptive, and can be adjusted to current data by means of training. Third, the model is content-aware, i.e. depends on the data on which the encoder operates. The content dependence is captured by means of visual clues, a measure of content complexity. Specifically, we use the texture and the motion complexity as the visual clues. Finally, since
the same model is used by sequence- and the frame-level controllers, the predictors are linear and can be
applied on data units of different granularity (frame, region of macroblocks or macroblock). Specifically for
the sequence-level controllers, we are interested in frame-wise predictors, which are obtained by summing
up or averaging the macroblock-wise predictors.

In general, if \( x \) is the quantity to be predicted (amount of bits, distortion, time) for the current frame, the
predicted value according to the encoder model is computed by

\[
\hat{x} = K \sum_{i=1}^{N_{MB}} \hat{x}_i(\theta_i; z_i)
\]  

(3)

where \( \hat{x}_i \) is a macroblock-wise predictor, \( \theta_i \) are the encoding parameters and \( z_i \) are the visual clues in macro-
block \( i \), \( N_{MB} \) is the number of macroblocks in the frame and \( K \) is some normalization factor. Specifically,
we use the following models:

**Bit production model:** in each macroblock, the amount of bits produced is proportional to the quantization
step \( q_i^{-1} \) and to the macroblock complexity \( z_i \). Depending on the frame type, either texture complexity
(in I-frames) or motion complexity (in P-frames) are used. The macroblock-wise predictor is given by

\[
\hat{b}_i(q_i, z_i) = \alpha_1 + \alpha_2 z_i q_i^{-1},
\]  

(4)

where \( \alpha_1, \alpha_2 \geq 0 \) are the model parameters obtained by training. Assuming that \( q \) is constant for the
entire frame, the frame-wise predictor of the amount of bits can be therefore expressed as

\[
\hat{b} = \sum_{i=1}^{N_{MB}} \hat{b}_i(q_i, z_i) = N_{MB} \alpha_1 + \sum_{i=1}^{N_{MB}} \alpha_2 z_i q_i^{-1}
\]  

(5)

\[
= \alpha^b + \beta^b q^{-1},
\]

where the coefficients \( \alpha^b = N_{MB} \alpha_1 \) and \( \beta^b = \sum_{i=1}^{N_{MB}} \alpha_2 z_i \) depend on the frame content and frame
type.

**Distortion model:** we use luma PSNR as an objective distortion criterion. In each macroblock, the PSNR
is inversely proportional to the quantization parameter \( q_i' \) and to the macroblock complexity \( z_i \) (texture
complexity is used for both P- and I-frames).

\[
\hat{b}_i(q_i, z_i) = \beta_1 - \beta_2 z_i - \beta_3 q_i',
\]  

(6)
where $\beta_1, \beta_2, \beta_3 \geq 0$ are the model parameters obtained by training. Assuming that $q'$ is constant for the entire frame, the predictor of the frame-wise average PSNR can be therefore expressed as

$$\hat{p} = \frac{1}{N_{MB}} \sum_{i=1}^{N_{MB}} \hat{p}_i(q', z_i) = \beta_1 - \frac{1}{N_{MB}} \sum_{i=1}^{N_{MB}} \beta_2 z_i - \frac{1}{N_{MB}} \sum_{i=1}^{N_{MB}} \beta_3 q'$$

(7)

where the coefficients $\alpha^p = \beta_1 - \frac{1}{N_{MB}} \sum_{i=1}^{N_{MB}} \beta_2 z_i$ and $\beta^p = -\frac{1}{N_{MB}} \sum_{i=1}^{N_{MB}} \beta_3$ depend on the frame content and frame type.

**Distortion model for dropped frames:** In some cases, the sequence-level controller may decide to drop a frame. In such a case, it is assumed that the decoder delays the previous frame for a longer duration, equal to the duration of the previous and current (dropped) frame. The distortion due to the drop is computed as the average PSNR between a downsampled (16 times in each axis) version the dropped frame and the previous frame. In our implementation, the average PSNR is computed by first calculating the average MSE and then converting it into logarithmic scale (rather than averaging the PSNR values directly). For example, dropping a frame at the beginning of a new shot would result in a low PSNR, while dropping a frame in a sequence of frames with slow motion will result in high PSNR. It may appear that though the average PSNR is high, in some small portions of the image the distortion due to frame drop is very large (such a situation is typical for a frame with abrupt motion of a foreground object, while the background is static). In order to take into consideration such situations, together with the average PSNR, we measure the 0.1% lowest quantile of the PSNR values, denoted by $\hat{p}^{\min}$.

**Time complexity model:** The underlying assumption of our time complexity model is that the encoding pipeline used to encode a macroblock consists of multiple stages, some of which are bypassed according to the encoded macroblock type, using some decision tree. Consequently, it can be simplistically assumed that encoding a macroblock of certain type takes a fixed amount of time. Let $cT$ denote the maximum amount of time it takes to encode a macroblock (nominal encoding time), which depends on the frame type and on the complexity scale $c$. We denote by $\tau'_i$ the encoding time assuming, i.e., the fraction of $cT$ is taken to encode macroblock $i$, such that $\tau_i = cT \tau'_i$. According to our model, the
normalized encoding time for macroblock $i$ is given by

$$
\hat{\tau}'_i(q_i, z_i) = \max\{\min\{\gamma_1 + \gamma_2 z_i - \gamma_3 q'_i, 1\}, 0\}T,
$$

where $\gamma_1, \gamma_2, \gamma_3 \geq 0$ are the model parameters obtained by training, and $z_i$ are the texture complexity (for I-frames) or motion complexity (for P-frames). The minimum and maximum is taken in order to ensure that the predicted normalized time value is within a valid range of $[0, T]$ (for a reasonable range of $z_i$ and $q'_i$, this assumption will hold). Assuming that $q'$ is constant for the entire frame, the predictor of the encoding time for the entire frame is approximated (neglecting the possible nonlinearity due to the minimum and maximum functions) as

$$\hat{\tau} \approx \sum_{i=1}^{N_{MB}} \hat{\tau}'_i(q'_i, z_i)c = N_{MB}\gamma_1 cT - \sum_{i=1}^{N_{MB}} \gamma_2 z_i cT - N_{MB}\beta cT q'_i
$$

where the coefficients $\alpha^c = N_{MB}\gamma_1 T - \sum_{i=1}^{N_{MB}} \gamma_2 z_i T$ and $\beta^c = -N_{MB}\beta cT$ depend on the frame content and frame type.

5 Buffer models

The purpose of the buffer models is to provide a simple means of modeling the encoder input and output as well as the hypothetic decoder input behavior, imposing constrains in bit and time allocation. The encoder model comprises a raw frame buffer constituting the input of the system to which raw frames are written, and the encoder bit buffer, constituting the output of the system, to which the encoded bitstream is written. The hypothetic decoder model comprises a decoder bit buffer, connected to the encoder bit buffer by a channel responsible for the transport of the coded stream.

5.1 Raw frame buffer model

The fullness of the raw frame buffer at time $t$ is denoted by $\ell^{raw}(t)$. We assume that initially $\ell^{raw}(0) = 0$, and raw frames are being filled at constant time intervals of $1/F$, where $F$ denotes the input sequence frame rate. Though $\ell^{raw}$ may assume integer values only, in most parts of our treatment we will relax this restriction treating $\ell^{raw}$ as a continuous quantity.
The encoder starts reading the first frame from the raw buffer as soon as the raw buffer level reaches \(\ell_{\text{raw}} \geq \ell_{\text{raw, min}}\), where \(\ell_{\text{raw, min}} > 1\) denotes the minimum amount of lookahead required by the encoder in our implementation. The first frame in the sequence starts being encoded at the time

\[
t_0 = \frac{\ell_{\text{min}}}{F}.
\]

(10)

The encoding of the first frame takes \(\tau_1\) seconds, after which the frame is removed from the raw buffer, decreasing its level by 1. The moment this happens is the encoding end time, denoted by

\[
\tau_{\text{enc}} = t_0 + \tau_1.
\]

(11)

If \(\ell_{\text{raw}}(t_{\text{enc}}) + \epsilon < \ell_{\text{raw, min}}\), for infinitesimal \(\epsilon > 0\), encoder raw buffer underflow occurs; in this case the encoder will stall until the minimum buffer level is reached. We denote this idle time by

\[
\tau_{\text{idle}} = \frac{\ell_{\text{min}} - \ell_{\text{raw}}(t_{\text{enc}})}{F}.
\]

(12)

In this notation, the encoding of the next frame will start at time \(t_{\text{enc}} + \tau_{\text{idle}} = t_0 + \tau_1 + \tau_{\text{idle}}\). The raw frame buffer is assumed to be capable of holding up to \(\ell_{\text{raw, max}}\) raw frames; when the buffer capacity reaches this limit an encoder raw buffer overflow occurs, and the input process stalls. Whenever the duration of such a stall is not negligible, input data may be lost.

The buffer level has to be checked for overflow immediately before the encoder finishes encoding a frame, and for underflow immediately before the encoder starts encoding a frame. Combining the contributions of frame production by the input subsystem and consumption by the encoder, the raw frame buffer levels are given by

\[
\ell_{\text{raw}}(t_{\text{enc}} - \epsilon) = t_{\text{enc}} F - n + 1,
\]

(13)

as depicted in Figure 2 (first row).

### 5.2 Encoder bit buffer model

The fullness of the encoder bit buffer at time \(t\) is denoted by \(\ell_{\text{enc}}(t)\). We assume that initially \(\ell_{\text{enc}}(0) = 0\) and at the time \(t_{\text{enc}}\) when the encoder completes encoding frame \(n\), \(b_n\) bits corresponding to the access unit of the coded frame are added to the buffer. After the first time the buffer level exceeds \(\ell_{\text{init}}\), the buffer
starts being drained at a rate $r(t) \leq r_{\text{max}}(t)$, determined by the transport subsystem. If at a given time the buffer is empty, the instantaneous draining rate drops to $r(t) = 0$. This situation is referred to as encoder bit buffer underflow. Except for inefficiency of utilizing the full channel capacity, encoder buffer underflow poses no danger to the system. The complement situation of encoder bit buffer overflow occurs when $\ell_{\text{enc}}(t_{n-1}^{\text{enc}}) + b_n \geq \ell_{\text{enc}}^{\text{tot}}$. As a consequence, the encoder will stall until the bit buffer contains enough space to accommodate the encoded bits. If the stall lasts for a non-negligible amount of time, unpredictable results (including input raw frame buffer overflow) may occur.

The buffer level has to be checked for underflow immediately before the encoder finishes encoding a frame, and for overflow immediately after the encoding is finished. Combining the contributions of bit production by the encoder and consumption by the transport subsystem, the encoder bit buffer levels are given by

\begin{align}
\ell_{\text{enc}}(t_{n-1}^{\text{enc}} - \epsilon) &= \ell_{\text{enc}}(t_{n-1}^{\text{enc}} + \epsilon) - \int_{t_{n-1}^{\text{enc}}}^{t_{n}^{\text{enc}}} r(t)dt \quad (14) \\
\ell_{\text{enc}}(t_{n}^{\text{enc}} + \epsilon) &= \ell_{\text{enc}}(t_{n}^{\text{enc}} - \epsilon) + b_n, \quad (15)
\end{align}

as depicted in Figure 2 (second row). For convenience, let us denote by

\begin{equation}
\frac{1}{t_{n}^{\text{enc}} - t_{n-1}^{\text{enc}}} \int_{t_{n-1}^{\text{enc}}}^{t_{n}^{\text{enc}}} r(t)dt
\end{equation}

the average rate while the frame $n$ being encoded. Assuming that $r(t) = r_{\text{max}}(t)$ if $\ell_{\text{enc}}(t) > 0$ and $r(t) = 0$ otherwise, we obtain

\begin{equation}
r_n = \frac{1}{t_{n}^{\text{enc}} - t_{n-1}^{\text{enc}}} \min \left\{ \ell_{\text{enc}}(t_{n-1}^{\text{enc}} + \epsilon), \int_{t_{n-1}^{\text{enc}}}^{t_{n}^{\text{enc}}} r_{\text{max}}(t)dt \right\}. \quad (17)
\end{equation}

In these terms, we may write

\begin{equation}
\ell_{\text{enc}}(t_{n}^{\text{enc}} - \epsilon) = \ell_{\text{enc}}(t_{n-1}^{\text{enc}} + \epsilon) - (t_{n}^{\text{enc}} - t_{n-1}^{\text{enc}})r_n. \quad (18)
\end{equation}

### 5.3 Decoder bit buffer model

Bits drained from the encoder bit buffer at time $t$ at rate $r(t)$ are transported by the channel and are appended to the decoder bit buffer at time $t + \delta_{\text{chan}}$ at the same rate. The channel delay $\delta_{\text{chan}}$ need not to be constant;
in this case, maximum channel delay may be assumed. The fullness of the decoder bit buffer at time $t$ is denoted by $\ell_{\text{dec}}(t)$. Initially, $\ell_{\text{dec}}(0) = 0$; the buffer level remains zero until $t = t_{\text{enc}}^1 + \delta_{\text{chan}}$, where the first bits of the first frame start arriving. The decoder remains idle until $\ell_{\text{dec}} \geq \ell_{\text{init}}$, once the bit buffer is sufficiently full, the decoder removes $b_1$ bits from the buffer and starts decoding the first frame of the sequence. The time lapsing between the arrival of the first bit until the first access unit is removed from the bit buffer is given by the smallest $\delta_{\text{dec}}$ satisfying

$$\int_{t_{\text{enc}}^1}^{t_{\text{enc}}^1 + \delta_{\text{dec}}} r(t') dt' \geq \ell_{\text{init}}. \quad (19)$$

We denote by

$$t_{\text{dec}}^1 = t_{\text{enc}}^1 + \delta_{\text{chan}} + \delta_{\text{dec}} \quad (20)$$

the time when the first frame starts being decoded. The delay passing between production and consumption of the access unit corresponding to a frame is denoted by

$$\delta = \delta_{\text{chan}} + \delta_{\text{dec}}. \quad (21)$$

The decoder removes access units corresponding to the encoded frames at a constant rate $F$, resulting in the following decoding schedule

$$t_{\text{dec}}^n = t_{\text{dec}}^{n-1} + \frac{n-1}{F}. \quad (22)$$

The decoder bit buffer level assumes highest and lowest values immediately before frame decoding start, and immediately after the decoding is started, respectively. Combining the contributions of bit production by the transport subsystem and consumption by the decoder, the decoder bit buffer levels are given by

$$\ell_{\text{dec}}(t_{\text{dec}}^n - \epsilon) = \ell_{\text{dec}}(t_{\text{dec}}^{n-1} + \epsilon) + \int_{t_{\text{dec}}^{n-1} + \delta}^{t_{\text{dec}}^n} r(t) dt \quad (23)$$

$$\ell_{\text{dec}}(t_{\text{dec}}^n + \epsilon) = \ell_{\text{dec}}(t_{\text{dec}}^n - \epsilon) - b_n, \quad (24)$$

as depicted in Figure 2 (third row).
6 Controller architecture

The sequence-level controller flow diagram is depicted in Figure 3. The encoder looks ahead for \( n \) frames and decides the control sequence for their encoding based on the current state of the buffers and the channel.

1. **Encoding order optimization** (optional, Section 10) is invoked to find the best order \( \pi^* \) of encoding the set of frames currently residing in the raw frame buffer. Decoded picture buffer size constraints are imposed to rule out some of the orderings. For the remaining orderings, the order optimization algorithm invokes the frame type decision algorithm.

2. **Frame type decision** (Section 9) is invoked with a frame reordering \( \pi \) to find the best sequence \( \alpha^* \) of encoding parameters, which includes the frame types, quantizer values, and complexity scales. The buffer model is invoked to obtain the current status of the encoder and estimated decoder buffers. A branch-and-bound algorithm is employed to efficiently search through different frame type combinations. For every feasible frame type assignment \( f \), the sequence resource allocation algorithm is invoked.

3. **Sequence resource allocation** (Section 7) receives the current buffer status from the buffer model, and invokes the encoder model with the set of frame types \( f \) to obtain the content-dependent frame-level bit, PSNR, and encoding time model parameters \( \alpha^b, \beta^b, \alpha^p, \beta^p, \alpha^t, \beta^t \). The sequence resource allocation algorithm establishes the budgets \( b_T \) and \( \tau_T \) of bits and encoding time, respectively, for the sequence of \( n \) frames, and invokes the frame resource allocation algorithm.

4. **Frame resource allocation** (Section 8) receives the model parameters frame-level bit, PSNR, and encoding time model parameters \( \alpha^b, \beta^b, \alpha^p, \beta^p, \alpha^t, \beta^t \) and the budgets \( b_T \) and \( \tau_T \), and find the optimal sequences \( q^* \) and \( c^* \) of quantizer and complexity scales, respectively. Together with the sequence \( f \) of frame types, this provides the entire set \( \alpha \) of encoding parameters (up to an optional frame reordering \( \pi \)), required to encode the sequence of \( n \) frames.

5. **Frame encoding**: once the best sequence \( \alpha^* \) of encoding parameters is found and returned by the highest-level encoding order optimization algorithm, the first set \( \alpha^*_1 \) of control parameters is used to encode the frame \( \pi_1 \).

6. **Update**: once the encoder has finished encoding one frame, the actual encoding time and amount of produced bits, as well as the actual rate and future rate estimates are observed and used to update the buffer models. The encoded frame is discarded from the raw frame buffer, and the process starts over.
7 Sequence resource allocation

Assuming that the encoder is encoding an \(i\)-th frame in the current GOP, and is capable of observing \(n - 1\) additional future frames \(i + 1, \ldots, i + n - 1\) queued in the raw input buffer, the problem of sequence resource allocation consists of establishing a resource budget for the latter sequence of \(n\) frames. Here, we consider allocation of bit and encoding time budgets, denoted by \(b_T\) and \(\tau_T\), respectively. We denote the \(i\)-th frame’s display time stamp relative to the beginning of the current GOP by \(t_i\). Assuming an estimate \(\hat{\delta}\) of the decoder delay is available, we denote by

\[
  t_{\text{max}} = \max\{t_{\text{gop}}^{\text{max}}, \hat{\delta}\} \quad (25)
\]

the maximum GOP duration (if the duration is smaller than the estimated delay, \(t_{\text{max}} = \hat{\delta}\) is used). The minimum GOP duration is set to the minimum temporal distance between two consecutive IDR frames,

\[
  t_{\text{min}} = t_{\text{idr}}^{\text{min}}. \quad (26)
\]

The maximum amount of time remaining till the GOP end is denoted by

\[
  t_{\text{rem}} = t_{\text{max}} - t_i, \quad (27)
\]

from where the number of remaining frames

\[
  n_{\text{rem}} = t_{\text{rem}}^F \quad (28)
\]

is obtained.

7.1 Effective bit rate estimation

We denote by \(r\) the current rate, at which the encoder bit buffer is being drained, and by \(\hat{r}^{\text{dec}}\) the estimated rate at the decoder at the time when the \(i\)-th frame will be decoded.

\[
  \hat{r}^{\text{dec}} \approx r(t_{\text{enc}} + \delta). \quad (29)
\]

Assuming the simplistic model of the rate remaining constant \(r\) in the interval \([t_{\text{enc}}, t_{\text{enc}} + \delta]\), and then remaining constant \(\hat{r}^{\text{dec}}\), the average bit rate of the encoder buffer drain in the interval \([t_{\text{enc}}, t_{\text{enc}} + t_{\text{max}}]\) is
given by

$$r_{enc} = \hat{\delta}r + \frac{(t_{max} - \hat{\delta}) \hat{r}_{dec}}{t_{max}}. \quad (30)$$

Under the assumption of nearly constant channel rate, the encoder and decoder bit buffers remain synchronized (up to an initial transient), connected by the relation

$$\ell_{enc}(t) + \ell_{dec}(t + \delta) = \ell_{enc}^{tot}. \quad (31)$$

However, bit rate fluctuations result in a loss of such a synchronization. For example, in the case of a future decrease in the channel capacity ($\hat{r}_{dec} < r$), encoding the next frames at the rate $r$ will cause the decoder buffer level to drop and, eventually, to underflow, while keeping the encoder buffer level balanced. In the converse case of a future increase in the channel capacity ($\hat{r}_{dec} > r$), keeping the video stream rate at $r$ will result in a decoder bit buffer overflow.

A solution proposed here consists of modifying the rate at which the stream is encoded such that both buffers remain as balanced as possible. We refer to such a modified rate as the effective bit rate,

$$r_{eff} = \max\{\min\{0.5(r_{enc} + \hat{r}_{dec}) + \Delta r, \max\{r_{enc}, \hat{r}_{dec}\}\}, \min\{r_{enc}, \hat{r}_{dec}\}\}, \quad (32)$$

where

$$\Delta r = \frac{1}{2t_{max}} \begin{cases} \hat{r}_{dec} - \ell_{enc} + \ell_{enc} - \ell_{dec} : \hat{r}_{dec} \geq r \\ \ell_{dec} - \ell_{max} - \ell_{max} + \hat{r}_{dec} : \hat{r}_{dec} < r \end{cases} \quad (33)$$

Using $r_{eff}$ instead of $r$ in our first example will cause both encoder and decoder buffer levels to drop by a less extent.

### 7.2 Bit budget allocation

Given an estimated effective encoding bit rate $r_{eff}$, the amount of bit budget remaining till the end of the GOP is given by

$$b_{rem} = r_{eff}t_{rem}. \quad (34)$$
According to the bit production model (6), the estimated amount of bits generated by the encoder for an $i$-th frame given a quantizer $q_i$ is given by

$$\hat{b}_i = \alpha_i^b + \beta_i^b \frac{1}{q_i},$$

(35)

where the coefficients $\alpha^b = (\alpha_1^b, ..., \alpha_n^b)^T$ and $\beta^b = (\beta_1^b, ..., \beta_n^b)^T$ depend on the frame content and frame type $f_i$. The latter is assumed to be assigned by a higher-level frame type decision algorithm detailed in the sequel. We write

$$\alpha^b = \sum_{i=1}^{n} \alpha_i^b,$$

$$\beta^b = \sum_{i=1}^{n} \beta_i^b$$

(36)

to denote the sum of the model coefficients of the observed $n$ frames.\(^1\) Substituting $\alpha^b$ and $\beta^b$ into the bit production model yields the total estimate of bits produced by encoding the observed frames $i, ..., i + n - 1$ with a constant quantizer. Similarly, we denote by

$$\alpha_{\text{rem}}^b = (n_{\text{rem}} - n) \alpha^b,$$

$$\beta_{\text{rem}}^b = (n_{\text{rem}} - n) \beta^b$$

(37)

the model coefficients for estimating the amount of bits of the remaining frames in the current GOP; $\alpha^b$ and $\beta^b$ are decaying moving averages of $\alpha_i^b$ and $\beta_i^b$, respectively, of the previously encoded frames.

The goal of sequence bit budget allocation is to maintain a fair distribution of the encoding bits throughout the sequence, considering the long-term effect of the decided allocation, as well as reacting to changes in channel bit rate and frame texture and motion complexity. Under the simplifying assumption that at the sequence level the visual quality is a function of the quantizer only, not depending on the frame content, the problem of sequence bit budget allocation can be translated to finding a single quantization scale $q$, which assigned to the remainder of the GOP frames produces $b_{\text{rem}}$ bits. Substituting the appropriate bit production

\(^1\)If $i$ is an IDR frame, the contribution of $\alpha_i^b$ and $\beta_i^b$ to the above sums can be increased by a multiplicative factor of $(1 + \gamma^b)$, where $\gamma^b \in [0, 1]$. This ensures that more bits are granted to IDR frames, thus resulting in a slight quality boosting at the beginning of each GOP.
models,

\[ \alpha^b + \alpha^b_{\text{rem}} + \left( \beta^b + \beta^b_{\text{rem}} \right) \frac{1}{q} = b_{\text{rem}} \]  

(38)

and solving for \(1/q\) yields

\[ \frac{1}{q} = \frac{b_{\text{rem}} - \alpha^b + \alpha^b_{\text{rem}}}{\beta^b + \beta^b_{\text{rem}}} \].  

(39)

Substituting the latter result into the bit production model yields the following bit budget for the sequence of the observed \(n\) frames

\[ b_T' = s \cdot \begin{cases} 
\alpha^b + \beta^b_{\text{rem}} \cdot \frac{b_{\text{rem}} - \alpha^b + \alpha^b_{\text{rem}}}{\beta^b + \beta^b_{\text{rem}}} & : \beta^b + \beta^b_{\text{rem}} > 0 \\
\frac{b_{\text{rem}}}{b_{\text{rem}}^\text{dec}} & : \text{else}, 
\end{cases} \]  

(40)

where

\[ s = \frac{\hat{\ell}_{\text{dec}}^\text{tot} + (\frac{\epsilon_1}{h} - \epsilon_1 - 1) \hat{\ell}_{\text{dec}}}{\hat{\ell}_{\text{dec}}^\text{tot} + (\frac{\epsilon_1}{h} - \epsilon_1 - 1) \hat{\ell}_{\text{dec}}'} \]  

(41)

is a scaling factor decreasing the bit budget for higher, and increasing for lower decoder bit buffer levels, respectively; \(\hat{\ell}_{\text{dec}}' = \hat{\ell}_{\text{dec}}(t_{\text{dec}} - \epsilon)\) is the estimated decoder bit buffer fullness immediately prior to decoding the \(i\)-th frame, \(\epsilon_1 = 3\), and \(h\) is the target relative decoder bit buffer level, defined as

\[ h = \max \left\{ \min \left\{ 0.5 + 0.1 \frac{t_i - t_{\text{min}}}{t_{\text{max}} - t_{\text{min}}} + t_{\text{max}} \frac{\hat{\ell}_{\text{dec}}' - r_{\text{eff}}}{\hat{\ell}_{\text{dec}}^\text{tot}}, 0.6 \right\}, 0.4 \right\} . \]  

(42)

To guarantee some level of bit allocation fairness, the budget is constrained to be within 25\% to 200\% of the average bit budget,

\[ \overline{b_T} = \frac{n r_{\text{eff}}}{F}, \]  

(43)

resulting in

\[ b_T'' = \min \{ \max \{ b_T', 0.25 \overline{b_T} \}, 2 \overline{b_T} \} . \]  

(44)
Encoder and decoder bit buffer constrains are further imposed, yielding the following final expression for the sequence bit budget

\[
b_T = \min \left\{ \min \left\{ b'_T, \ell_{\text{enc}} - \ell_{\text{enc}}^\prime + \frac{n r}{F} \right\}, \ell_{\text{dec}} + n \frac{r_{\text{dec}}}{F} - \ell_{\text{dec}}^\prime \right\},
\]

(45)

where \( \ell_{\text{enc}}^\prime \) is the encoder bit buffer level immediately prior to encoding the \( i \)-th frame.

### 7.3 Encoding time budget allocation

Similar to the bit budget allocation, the time budget allocation problem consists of assigning the target encoding time \( \tau_T \) for the observed frames \( n \). However, due to the stricter constrains imposed by the encoder raw frame buffer, encoding time allocation operates with shorter terms. Since frames are added to the encoder input buffer at the rate of \( F \) frames per second, the average encoding time of a frame is \( \frac{1}{F} \), resulting in the average budget

\[
\tau_T = \frac{n}{F}
\]

(46)

for the sequence of \( n \) observed frames. Using this budget and assuming the encoder has encoded \( n_{\text{enc}} \) frames so far (including the dropped ones), the time at the encoder should be \( n_{\text{enc}}/F \). However, since the actual encoding time may differ from the allocated budget, the time at the encoder immediately prior to encoding the \( i \)-th frame (relative to the time \( t_0 \) when the first frame in the sequence starts being encoded) usually differs from the ideal value. We denote by

\[
t_{\text{dif}} = n_{\text{enc}} \frac{1}{F} - \ell_{\text{enc}} - t_0
\]

(47)

the time difference between the ideal and actual encoding time; if \( t_{\text{dif}} > 0 \) the encoder is faster than the input raw frame rate and the encoding time budget has to be increased in order to avoid raw buffer underflow. Similarly, if \( t_{\text{dif}} < 0 \), the encoder is lagging behind the input and the time budget has to be decreased in order to prevent an overflow. Demanding the encoder to close the time gap in \( n_{\text{resp}} \) frames \( (n_{\text{resp}}/F \) seconds), yields the following encoding time budget

\[
\tau'_T = \tau_T + \frac{n t_{\text{dif}}}{n_{\text{resp}}},
\]

(48)
Typically, \( n_{\text{resp}} \approx 5 \), depending on \( \ell_{\text{max}} \). To guarantee some level of fairness, the budget is constrained by

\[
\tau_T'' = \max\{\min\{\tau_T', 1.5\tau_T\}, 0.75\tau_T\}.
\]

Encoder bit buffer constrains are further imposed yielding the final encoding time budget

\[
\tau_T = \max\{\min\{\tau_T'', \tau_{\text{max}}\}, \tau_{\text{min}}\},
\]

where

\[
\tau_{\text{min}} = \ell_{\text{enc}} + \sum_{i=1}^{n} \hat{b}_i - \ell_{\text{max}},
\]

\[
\tau_{\text{max}} = \max\left\{\frac{\ell_{\text{enc}} + \sum_{i=1}^{n-1} \hat{b}_i - \ell_{\text{enc}}}{r}, \tau_{\text{min}}\right\},
\]

and \( \hat{b}_i \) are the expected bits produced by encoding the \( i \)-th frame. Due to the dependence on \( \hat{b}_i \), the time allocation must be preceded by bit allocation.

8 Frame resource allocation

The problem of frame resource allocation consists of distributing a given budget of resources between \( n \) frames in a sequence. More formally, we say that the vector \( \mathbf{x} = (x_1, ..., x_n)^T \) is an allocation of a resource \( x \), where each \( x_i \) quantifies the amount of that resource allocated to the frame \( i \). For example, \( x \) can represent the amount of coded bits or encoding time. Ideally, we would like to maximize the \( x_i \) for each frame; however, the allocation has to satisfy some constrains, one of which is

\[
\sum_{i=1}^{n} x_i = x_T,
\]

where \( x_T \) is the resource budget assigned to the sequence by a higher-level sequence resource allocation algorithm. Other constrains stemming, for example, from the encoder buffer conditions apply as well. Formally, we say that an allocation vector \( \mathbf{x} \) is feasible if it satisfies that set of conditions, and infeasible otherwise. Resource allocation can be therefore thought of as finding a feasible solution to the maximization
problem

$$\max_{\mathbf{x}} \mathbf{x} \quad \text{s.t.} \quad \mathbf{x} \text{ is feasible}. \quad (53)$$

Since usually the encoder is controlled using a set of parameters different from the allocated resource itself (e.g., though coded bits are allocated as a resource, the amount of produced bits is controlled through the quantization parameter), it is more convenient to express $\mathbf{x}$ as a function of some vector of parameters $\mathbf{\theta} = (\theta_1, ..., \theta_m)^T$. This yields the following optimization problem

$$\max_{\mathbf{\theta}} \mathbf{x}(\mathbf{\theta}) \quad \text{s.t.} \quad \mathbf{x}(\mathbf{\theta}) \text{ is feasible}. \quad (54)$$

Note, however, that since the maximized objective is a vector rather than a scalar quantity, there exists no unique way to define what is meant by “maximum $\mathbf{x}$”. Here, we adopt the following notion of vector objective optimality.

**Definition 1** (Max-min optimality). A vector of resources $\mathbf{x} = (x_1, ..., x_n)^T$ is said to be max-min optimal if it is feasible, and for any $1 \leq i \leq n$ and a feasible $\mathbf{y} = (y_1, ..., y_n)^T$ for which $x_p < y_p$, there is some $j$ with $x_i \geq y_j$ and $x_j > y_j$.

Informally, this means that it is impossible to increase the resource allocated to a frame $i$ without decreasing the resources allocated to frames that have already a smaller resource allocation than $x_i$, and without violating the feasibility constraints. For this reason, a max-min optimal resource allocation can be thought of as fair. As to notation, given the vector of resources $\mathbf{x}$ as a function of some parameters $\mathbf{\theta}$ and feasibility constraints $\mathbf{\theta} \in \Omega$, we will henceforth interpret the vector objective maximization problem

$$\mathbf{\theta}^* = \arg \max_{\mathbf{\theta} \in \Omega} \mathbf{x}(\mathbf{\theta}), \quad (55)$$

as finding a max-min optimal resource allocation $\mathbf{x}^* = \mathbf{x}(\mathbf{\theta}^*)$. In the remainder of this section, we are going to explore and formulate allocation problems for two types of resources considered here, namely coded bits and encoding time.

### 8.1 Bit allocation

The first problem we are going to address is the distribution of the budget of $b_T$ bits between the frames, which can be denoted as a bit allocation vector $\mathbf{b} = (b_1, ..., b_n)^T$. However, since the encoder is not
controlled directly by \( b \), but rather by the quantizer value, we reformulate the problem as allocating a vector \( q = (q_1, \ldots, q_n)^T \) of quantizer scales (or, equivalently, a vector \( q' \) of quantization parameters). Though \( q \) and \( q' \) may assume only discrete values, we relax this restriction by making them continuous variables. In these terms, the quantizer allocation problem can be formulated as max-min allocation of coding quality, in our case, average frame PSNR values \( p = (p_1, \ldots, p_n)^T \), as function of \( q \) and subject to feasibility constrains,

\[
\max_{q} p(q) \quad \text{s.t.} \quad b(q) \text{ is feasible.} \tag{56}
\]

Note that the feasibility constrains are imposed on the amount of bits produced by the encoder with the control sequence \( q \). Since in practice the exact amount of bits produced for a given quantizer is unknown \textit{a priori}, \( b(q) \) has to be substituted with the estimated coded bits \( \hat{b}(q) \). Such an estimate depends on the frame types, which are assumed to be decided by a higher-level frame type decision algorithm, detailed in the sequel.

One of the feasibility constrains is, clearly, \( \hat{b}(q) = b_T \). However, this condition is insufficient, as it may happen that the produced bits violate the buffer constraints for a frame \( i < n \). In order to formulate the buffer constrains on the allocated bits, let us denote by \( t_{\text{dec}} = (t_{\text{dec}}^1, \ldots, t_{\text{dec}}^n)^T \) the decoding timestamps of the frames, assuming without loss of generality that \( t_1 = 0 \) and that the frames are numbered in the increasing order of \( t_i \). We denote the estimated decoder bit buffer fullness immediately before and immediately after frame \( i \) is decoded by \( \hat{\ell}_{\text{dec}}^{i-} = \hat{\ell}_{\text{dec}}^{i} (t_{\text{dec}}^i - \epsilon) \) and \( \hat{\ell}_{\text{dec}}^{i+} = \hat{\ell}_{\text{dec}}^{i} (t_{\text{dec}}^i + \epsilon) \), respectively. We assume that the initial decoder buffer fullness \( \hat{\ell}_{\text{dec}}^{1-} = \ell_{\text{dec}}^0 \) is given as the input. In vector notation, we can write

\[
\begin{align*}
\hat{\ell}_{\text{dec}}^{i-}(q) &= \ell_{\text{dec}}^0 - \mathbf{K} \hat{b}(q) + \hat{\ell}_{\text{dec}}^{\text{dec}}_{\text{dec}} \\
\hat{\ell}_{\text{dec}}^{i+}(q) &= \ell_{\text{dec}}^0 - \mathbf{J} \hat{b}(q) + \hat{\ell}_{\text{dec}}^{\text{dec}}_{\text{dec}}
\end{align*}
\tag{57}
\]

where

\[
\mathbf{J} = \begin{pmatrix}
1 & 0 & \cdots & 0 \\
1 & 1 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 1 & \cdots & 1
\end{pmatrix}, \tag{59}
\]

\( \mathbf{K} = \mathbf{J} - \mathbf{I} \), and \( \hat{\ell}_{\text{dec}}^{\text{dec}} \) is the estimated average decoder bit buffer filling rate on the time interval \( [t_1, t_n + \frac{1}{F}] \).
The constrained optimal allocation problem becomes

\[
\max_{q_{\min} \leq q \leq q_{\max}} p(q) \quad \text{s.t.} \quad \begin{cases} 
1^T \hat{b}(q) = b_T \\
\ell_{\text{dec}}^- (q) \leq \ell_{\text{dec}} \leq \ell_{\text{dec}}^+ (q) \geq \ell_{\text{dec}, \min}.
\end{cases} \tag{60}
\]

where \(\ell_{\text{dec}, \min}\) and \(\ell_{\text{dec}, \max}\) are the minimum and the maximum decoder bit buffer levels, respectively. Since it is reasonable to assume a monotonically decreasing dependence of the PSNR in the quantization parameter (or the quantizer scale), the maximizer of (60) coincides with the minimizer of

\[
\min_{q_{\min} \leq q \leq q_{\max}} q \quad \text{s.t.} \quad \begin{cases} 
1^T \hat{b}(q) = b_T \\
\ell_{\text{dec}}^- (q) \leq \ell_{\text{dec}} \leq \ell_{\text{dec}}^+ (q) \geq \ell_{\text{dec}, \min}.
\end{cases} \tag{61}
\]

A numerical solution of the max-min allocation problem (61) can be carried out using a variant of the bottleneck link algorithm, summarized in Algorithm 1. In the algorithm, we assume that the amount of coded bits produced for a frame \(i\) as a function of \(q_i\) is given by the model

\[
\hat{b}_i = \alpha_i b + \beta_i q_i, \tag{62}
\]

where the coefficients \(\alpha_i^b = (\alpha_{i,1}^b, ..., \alpha_{i,n}^b)^T\) and \(\beta_i^b = (\beta_{i,1}^b, ..., \beta_{i,n}^b)^T\) depend on the frame content and frame types. In vector notation, this yields

\[
\hat{b} = \alpha^b + \beta^b q, \tag{63}
\]

where vector division is interpreted as an element-wise operation. The algorithm can be easily adopted to other models as well by replacing the closed-form solution in Steps 1, 8, and 9 with the more general equation

\[
b_T = \sum_{i=1}^n \hat{b}_i(q) \tag{64}
\]

with respect to the scalar \(q\).
Find optimal equal quantizer allocation
\[ q = \max \left\{ \min \left\{ \frac{1^T \beta}{b_T - 1^T \alpha}, q_{\text{max}} \right\}, q_{\text{min}} \right\} \]
and set \( q = q \cdot 1 \).

2 if \( \ell_{\text{dec}}^-, (q) \leq \ell_{\text{dec}}^\text{max} \) and \( \ell_{\text{dec}}^+, (q) \geq \ell_{\text{dec}}^\text{min} \) then
   \[ \text{Set } q^* = q. \]
3 else
   Find the smallest \( 1 \leq m \leq n \) for which one of the constraints is violated.
   if \( \ell_{\text{dec}}^- (q) \leq \ell_{\text{dec}}^\text{max} \) then set \( b_0^T = \ell_0^\text{dec} + \hat{r} \cdot t_{\text{dec}} - \ell_{\text{dec}}^\text{max} \) else set \( b_0^T = \ell_0^\text{dec} + \hat{r} \cdot t_{\text{dec}} - \ell_{\text{dec}}^\text{min} \).
   Set \( b_1^T = \max \{ b_0^T, 0 \} \).
   Find
   \[ q = \max \left\{ \min \left\{ \frac{1^T (\beta_{m+1}^b, ..., \beta_n^b)^T}{b_1^T - 1^T (\alpha_{m+1}^b, ..., \alpha_n^b)^T}, q_{\text{max}} \right\}, q_{\text{min}} \right\} \]
   and set \( q_1^* = \cdots = q_m^* = q. \)
9 Compute remaining bit budget
   \[ b_T = \max \left\{ b_T - 1^T (\alpha_1^b, ..., \alpha_m^b)^T - \frac{1}{q} \cdot 1^T (\beta_{m+1}^b, ..., \beta_n^b)^T, 0 \right\} \]
10 Recursively invoke the algorithm with \( \alpha = (\alpha_{m+1}^b, ..., \alpha_n^b)^T, \beta^b = (\beta_{m+1}^b, ..., \beta_n^b)^T, \)
   \( \ell_{\text{dec}} = (t_{m+1}^\text{dec}, ..., t_n^\text{dec})^T - t_{\text{dec}}^\text{max}, \) and \( \ell_0^\text{dec} = \ell_0^\text{dec} - b_0^T + \hat{r} \cdot t_{\text{dec}}^m + \hat{r} \cdot t_{\text{dec}}^m \) to fix the remaining quantizer values \( (q_{m+1}^*, ..., q_n^*). \)
11 end

Algorithm 1: Max-min frame bit allocation.

8.2 Encoding time allocation

The second problem we are going to address is the allocation of the encoding time budget \( \tau_T \) between the frames, which can be denoted as an encoding time allocation vector \( \tau = (\tau_1, ..., \tau_n)^T \). As in the case of bit allocation, since the encoder is not controlled directly by \( \tau \), but rather by the complexity scale, we
reformulate the problem as allocating a vector $c = (c_1, ..., c_n)^T$ of complexity scales. Again, we think of $c$ as of a continuous variable, though in practice it may be restricted to a set of discrete values. We denote by $\hat{t}^{\text{enc}} = (\hat{t}_1^{\text{enc}}, ..., \hat{t}_n^{\text{enc}})^T$ the time at which frame $i$’s encoding is complete, assuming without loss of generality that the first frame starts being encoded at time 0. Furthermore, we assume that the encoding end time for a frame $i$ coincides with the encoding start time for the frame $i + 1$. In this notation,

$$\hat{t}^{\text{enc}} = J\hat{\tau}(c),$$

where $\hat{\tau} = (\hat{\tau}_1, ..., \hat{\tau}_n)^T$ denotes the estimated encoding times.

Assuming the quantizer allocation has fixed the estimated amount of bits $\hat{b}$ produced by encoding each frame, our goal is to maximize the coding quality $p(c)$ subject to the encoder buffer constrains. The encoder buffer constrains include both the encoder bit buffer constrains, applying to the output bit buffer, and the encoder raw buffer constrains, applying to the input raw frame buffer. Similar to the bit allocation problem, we denote the estimated encoder bit buffer fullness immediately before and immediately after frame $i$’s encoding is complete by $\hat{\ell}^{\text{enc}}_i = \hat{\ell}^{\text{enc}}(\hat{t}_i^{\text{enc}} - \epsilon)$ and $\hat{\ell}^{\text{enc}}_{i+1} = \hat{\ell}^{\text{enc}}(\hat{t}_i^{\text{enc}} + \epsilon)$, respectively. The initial buffer level $\ell_0^{\text{enc}} = \ell^{\text{enc}}(0)$ is assumed to be known from a direct observation. Using this notation, the buffer levels are given by

$$\hat{\ell}^{\text{enc}}_-(c) = \ell_0^{\text{enc}} + J\hat{\tau}(c)$$

and

$$\hat{\ell}^{\text{enc}}_+(c) = \ell_0^{\text{enc}} - J\hat{\tau}(c)$$

In the same manner, $\hat{\ell}^{\text{raw}}_i$ and $\hat{\ell}^{\text{raw}}_{i+1}$ denote the estimated encoder raw frame buffer fullness immediately before and immediately after frame $i$’s encoding is complete. The initial buffer level at time 0 is denoted by $\ell_0^{\text{raw}}$ and is available from a direct observation. Since the filling rate of the input buffer is $F$, we have

$$\hat{\ell}^{\text{raw}}(\hat{t}_i^{\text{enc}} - \epsilon) = \ell_0^{\text{raw}} + \hat{\ell}^{\text{enc}}(\hat{t}_i^{\text{enc}} - \epsilon)$$

and

$$\hat{\ell}^{\text{raw}}(\hat{t}_i^{\text{enc}} + \epsilon) = \hat{\ell}^{\text{raw}}(\hat{t}_i^{\text{enc}} - \epsilon) - 1.$$
The buffer-constrained encoding complexity allocation problem can be expressed as

\[
\max_{c_{\min} \leq c \leq c_{\max}} c \quad \text{s.t.} \quad \begin{cases} 
1^T \hat{\tau}(c) = \tau_T \\
\hat{\ell}_{+}^{\text{enc}}(c) \leq \ell_{\text{enc}}^{\max} \\
\hat{\ell}_{-}^{\text{enc}}(c) \geq \ell_{\text{enc}}^{\min} \\
\hat{\ell}_{+}^{\text{raw}}(c) \leq \ell_{\text{raw}}^{\max} \\
\hat{\ell}_{-}^{\text{raw}}(c) \geq \ell_{\text{raw}}^{\min}
\end{cases}
\]  \quad (71)

However, this formulation suffers from two potential problems. The first drawback stems from our assumption that encoding end time for a frame \(i\) coincides with the encoding start time for the frame \(i+1\). For example, if the \(i\) frame is dropped and the input buffer level falls below \(\ell_{\text{raw}}^{\min}\), the encoder will stall until the minimum buffer level is reached. This will make \(\tau_i\) non-negligible, which in turn will require \(c_i\) to be very high (or even infinite if we assume that the nominal encoding time for a dropped frame is strictly zero). We overcome this problem by relaxing the constrain \(c \leq c_{\max}\). If some elements of the optimal complexity allocation vector \(c^*\) exceed the maximum complexity scale, we will say that the encoder yields a portion of its CPU time to other processes potentially competing with it over the CPU resources. This fact can be quantified by introducing a vector of CPU utilization \(\eta = (\eta_1, ..., \eta_n)^T\), where \(0 \leq \eta_i \leq 1\) expresses the fraction of the CPU time used by the encoder process in the time interval \([\hat{t}_{\text{enc}}^{i-1}, \hat{t}_{\text{enc}}^i], \hat{t}_{\text{enc}}^0 = 0\). Setting,

\[
\eta_i = \frac{c_{\max}}{c_i^*},
\]  \quad (72)

and \(c = \min\{c^*, c_{\max}\}\), the encoding complexity scale of frames with \(c_i^*\) exceeding \(c_{\max}\) will be set to \(c_{\max}\), and the utilizable CPU time will be lower than 100%.

The second difficulty in the allocation problem (71) stems from the fact that sometimes the input and output buffer constrains may be conflicting (or, more formally, the feasible region may be empty).

**Definition 2** (Ordered constrains). *Given an ordered \(m\)-tuple of indicator functions \(\chi = (\chi_1, ..., \chi_m)^T, \chi_i : \mathbb{R}^n \to \{0, 1\}, a vector \(x\) is said to satisfy the ordered constrains \(\chi(x)\) if there exists no \(y\) with \(\chi(y) \prec \chi(x)\), where \(\prec\) denotes the lexicographic order relation between binary strings.*

Informally, this definition implies that in case of conflicting constrains, satisfying the first constrain is more important than satisfying the second one, and so on. Using this relaxed notion of constrained optimality,
allocation problem (71) can be rewritten as

$$\max_{c \geq c_{\min}} c \quad \text{s.t.} \quad \begin{cases} \ell_{\text{raw}}^+ (c) \geq \ell_{\text{raw}} \min \vspace{0.2cm} \\
\ell_{\text{raw}}^- (c) \leq \ell_{\text{raw}} \max \\
\ell_{\text{enc}}^+ (c) \leq \ell_{\text{enc}} \max \\
\ell_{\text{enc}}^- (c) \geq \ell_{\text{enc}} \min \\
1^T \hat{\tau} (c) = \tau_T, \end{cases}$$

(73)

where the constraints are interpreted as ordered constraints.

The time complexity allocation (73) is solved using Algorithm 2 similar to Algorithm 1 for bit allocation. In the algorithm, we assume that the encoding time of a frame $i$ as a function of $c_i$ is given by the model

$$\hat{\tau}_i = \frac{\alpha_i^1 + \beta_i^1 q'_i c_i}{\eta_i} = \frac{\gamma_i c_i}{\eta_i},$$

(74)

where the coefficients and $\alpha^1 = (\alpha_1^1, ..., \alpha_n^1)^T$ and $\beta^1 = (\beta_1^1, ..., \beta_n^1)^T$ depend on the frame content and frame types, and $\gamma^1 = (\alpha_1^1 + \beta_1^1 q'_1, ..., \alpha_n^1 + \beta_n^1 q'_n)^T$.

### 8.3 Joint bit and encoding time allocation

Allocation problems (61) and (73) tacitly assume that allocation of quantization and encoder time complexity scales are independent. However, while it is reasonable to assume that the coding quality $p$ is a function of $q$ only (and, thus, is independent of $c$), the amount of produced bits $b$ clearly depends on both parameters. This dependence couples together the two problems through the more accurate expression for the encoder bit buffer levels

$$\ell_{\text{enc}}^- (q, c) = \ell_{0}^\text{enc} + K \hat{b}(q, c) - r J \hat{\tau} (c),$$

(75)

$$\ell_{\text{enc}}^+ (q, c) = \ell_{0}^\text{enc} + J \hat{b}(q, c) - r J \hat{\tau} (c),$$

(76)

and the decoder bit buffer levels

$$\ell_{\text{dec}}^- (q, c) = \ell_{0}^\text{dec} - K \hat{b}(q, c) + \hat{\tau} \text{dec} \ell_{\text{dec}},$$

(77)

$$\ell_{\text{dec}}^+ (q, c) = \ell_{0}^\text{dec} - J \hat{b}(q, c) + \hat{\tau} \text{dec} \ell_{\text{dec}},$$

(78)
input : \( n \times 1 \) vectors \( \gamma^t \) of time model parameters; initial encoder bit buffer level \( \ell^0_{\text{enc}} \); initial encoder raw frame buffer level \( \ell^0_{\text{raw}} \); time budget \( T \); measured average encoder buffer draining rate \( r \); frame rate \( F \).

output : \( n \times 1 \) optimal max-min complexity allocation vector \( c^* \); \( n \times 1 \) CPU time utilization vector \( \eta \).

1. Find optimal equal complexity scale allocation

\[
\hat{c} = \max \left\{ \frac{T}{1^T \gamma}, c_{\min} \right\}
\]

and set \( c = c \cdot 1 \).

2. if \( \ell^\text{enc}_+ (c) \leq \ell^\text{enc}_- (c) \) and \( \ell^\text{raw}_- (c) \geq \ell^\text{raw}_+ (c) \) then set \( c^* = c \)

else

3. Find the smallest \( 1 \leq m \leq n \) for which one of the constrains is violated.

4. Compute the complexity constrains

\[
c_1 = \frac{\ell^\text{max} - \ell^0_{\text{raw}} + m}{F \cdot (\gamma_1^t, \ldots, \gamma_m^t) 1} \quad c_2 = \frac{\ell^\text{min} - \ell^0_{\text{raw}} + m - 1}{F \cdot (\gamma_1^t, \ldots, \gamma_m^t) 1} \quad c_3 = \max \left\{ \min \left\{ \frac{\ell^\text{enc}_- (c) - (b_1, \ldots, b_m) 1}{r \cdot (\gamma_1^t, \ldots, \gamma_m^t) 1}, c_1 \right\}, c_1 \right\} \quad c_4 = \min \left\{ \max \left\{ \frac{\ell^\text{enc}_+ (c) - (b_1, \ldots, b_m - 1) 1}{r \cdot (\gamma_1^t, \ldots, \gamma_m^t) 1}, c_1 \right\}, c_2 \right\}
\]

and set \( c_1 = \max \{ c_1, c_3 \}, c_{\min} \} \) and \( c_2 = \min \{ c_2, c_4 \} \).

5. if \( \ell^\text{enc}_m (c) \leq \ell^\text{enc}_m (c) \) or \( \ell^\text{raw}_m (c) \geq \ell^\text{raw}_m (c) \) then set \( \tau_T^0 = c_{\min} (\gamma_1^t, \ldots, \gamma_m^t) 1 \) else set \( \tau_T^0 = c_{\max} (\gamma_1^t, \ldots, \gamma_m^t) 1 \).

6. Find

\[
c = \max \left\{ \frac{\tau_T^0}{(\gamma_1^t, \ldots, \gamma_m^t) 1}, c_{\min} \right\}
\]

and set \( c_1 = \ldots = c_m = c \).

7. Compute remaining time budget \( \tau_T = \max \{ \tau_T - c (\gamma_1^t, \ldots, \gamma_m^t) 1, 0 \} \).

8. Recursively invoke the algorithm with \( \gamma^t = (\gamma_1^t, \ldots, \gamma_m^t) 1 \), \( \ell_0^\text{enc} = \ell^\text{enc} (b_1, \ldots, \gamma_m) 1 - \tau_T^0 \)

and \( \ell_0^\text{raw} = \ell_0^\text{raw} + F (\gamma_1^t, \ldots, \gamma_m^t) 1 - m \) to fix the remaining complexity scale values \( c_{\max}^* \).

9. Set \( \eta = c_{\max}^* / c^* \) and \( c^* = \max \{ c^*, c_{\max} \} \).

10. end

**Algorithm 2**: Max-min frame time complexity allocation.
which now depend on both $q$ and $c$. Unfortunately, a joint bit and time complexity allocation problem is not well-defined, since combining the two vector-valued objectives $q$ and $c$ can no more be treated using the max-min optimality framework, as the two vectors are non-commensurable.\footnote{The possibility of using $p(q, c)$ as the new objective is also problematic, as $p$ depends on $q$ only.} However, joint allocation can be performed by alternatingly solving the bit and time allocation problems, as suggested by the following algorithm.

<table>
<thead>
<tr>
<th>initialization: Set $c^* = 1$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>repeat</td>
</tr>
<tr>
<td>Fix $c = c^<em>$ and find the optimal quantizer allocation $q^</em>$ by solving (61).</td>
</tr>
<tr>
<td>Fix $q = q^<em>$ and find the optimal time complexity allocation $c^</em>$ by solving (73).</td>
</tr>
<tr>
<td>until until convergence</td>
</tr>
</tbody>
</table>

Algorithm 3: Joint frame bit and encoding complexity allocation.

The convergence condition can be a bound on the change in $c^*$ and $q^*$, the number of iterations, or any combination thereof. Our practice shows that a single iteration of this algorithm usually produces acceptable allocation.

9 Frame type decision

The purpose of frame type decision is to associate with a sequence of $n$ frames an optimal sequence of frame types. The frame type with which an $i$-th frame is encoded is denoted by $f_i$. To simplify notation, we assume that $f_i$ also specifies whether the frame is used as reference, whether it is an IDR, and which frames are used for its temporal prediction (unless it is a spatially predicted frame). For example a frame $i = 1$ can be IDR, I, P predicted from frame 0, or DROP. The space of possible frame type assignment for a frame $i$ is denoted by $F_i$, and depends solely on the status of the reference buffer immediately before the frame is encoded, denoted by $R_i$ ($R_i$ is defined as the list of reference frame indices). Speaking more broadly, $F_i$ is a function of the encoder state immediately prior to encoding a frame $i$, defined as the pseudo-vector

$$\sigma_i = (R_i, \ell_{i-1}^{enc}, \ell_{i-1}^{raw}, \ell_{i-1}^{dec}, l_{i-1}^{nd}, t_{i-1}^{idr}),$$

(79)

where $\ell_{i-1}^{enc}$, $\ell_{i-1}^{raw}$, and $\ell_{i-1}^{dec}$ denote the levels of the encoder bit buffer, raw buffer, and decoder bit buffer, respectively, $l_{i-1}^{nd}$ denotes the index of the last non-dropped frame, and $t_{i-1}^{idr}$ denotes the presentation time of
the last IDR frame. $\sigma_i$ fully defines the instantaneous encoder state. In practice, only estimated buffer levels are available. We will henceforth denote the estimated encoder state by

$$\hat{\sigma}_i = (R_i, \hat{\ell}_i^\text{enc}, \hat{\ell}_i^\text{raw}, \hat{\ell}_i^\text{dec}, l_i^{nd}, t_i^{idr}). \quad (80)$$

Note that $R_i$, $l_i^{nd}$, and $t_i^{idr}$ in $\hat{\sigma}_i$ are fully deterministic.

It is important to observe that $f_i$ does not fully define the set of control parameters required by the encoder in order to encode the $i$-th frame, as it does not specify the quantization and complexity scales $q_i$ and $c_i$. In order to separate the optimization of frame type and reference indices from the optimal quantizer and complexity allocation, we assume that given the sequence of frame types $f = (f_1, ..., f_n)$, the allocation algorithms described in the previous section are invoked to find $q(f) = (q_1^*, ..., q_n^*)$ and $c(f) = (c_1^*, ..., c_n^*)$. As consequence, the amount of bits produced and the amount of time consumed by encoding the $i$-th frame can be expressed as functions of $f$. To simplify notation, we will denote the latter quantities by $b_i$ and $\tau_i$, respectively. Similarly, $p_i$ will denote the distortion of the $i$-th frame. In the case where $f_k = \text{DROP}$, the distortion is evaluated as the PSNR of the difference between the original frame $i$ and the last displayed frame $l_i^{nd}$.

We jointly refer to $a_i = (f_i, q_i^*, c_i^*)$ as the encoder action for the frame $i$. Note that although $a_i$ is defined for a single frame, the optimal values of $q_i^*$ and $c_i^*$ depend on the $f_i$’s of the entire sequence of frames 1, ..., $n$. As consequence, $a$ can be determined only as a whole, i.e. $a_{i+1}$ is required in order to determine $a_i$.

### 9.1 Encoder state update

Given the encoder state $\sigma_i$ and the action $a_i$, the next state $\sigma_{i+1}$ is unambiguously determined by the state update rule $\sigma_{i+1} = \sigma(\sigma_i, a_i)$. In practice, the update rule is applied to the estimated state, $\hat{\sigma}_{i+1} = \sigma(\hat{\sigma}_i, a_i)$. The update for the buffer levels is given by

$$\begin{align*}
\hat{\ell}_{i+1}^\text{enc} &= \hat{\ell}_i^\text{enc} + \hat{b}_i - r_i \hat{\tau}_i \\
\hat{\ell}_{i+1}^\text{raw} &= \hat{\ell}_i^\text{raw} + F \hat{\tau}_i - 1 \\
\hat{\ell}_{i+1}^\text{dec} &= \hat{\ell}_i^\text{dec} - \hat{b}_i + \hat{r}_i \text{dec}(t_i^{dec} - t_{i+1}^{dec}), \quad (81)
\end{align*}$$

where $t_i^{dec}$ is the decoding time stamp of the $i$-th frame, $r_i$ is the average encoder bit buffer draining rate at the time $t_i^{enc}$, and $\hat{r}_i$ is the predicted decoder bit buffer filling rate at the time $t_i^{dec}$. The last displayed frame
is updated according to
\[ t_{i+1}^{\text{nd}} = \begin{cases} t_i^{\text{nd}} : f_i = \text{DROP} \\ i : \text{else.} \end{cases} \tag{82} \]

The last IDR presentation time stamp is updated according to
\[ t_{i+1}^{\text{idr}} = \begin{cases} t_i : f_i = \text{IDR} \\ t_{i}^{\text{dr}} : \text{else.} \end{cases} \tag{83} \]

The reference buffer is updated according to the sliding window policy
\[ \mathcal{R}_{i+1} = \begin{cases} \emptyset : f_i = \text{IDR} \\ \mathcal{R}_i : f_i \in \text{NONREF} \\ \{i\} \cup \mathcal{R}_i : f_i \in \text{REF and } |\mathcal{R}_i| < \mathcal{R}_{\text{max}} \\ \{i\} \cup \mathcal{R}_i \setminus \{\min \mathcal{R}_i\} : \text{else}, \end{cases} \tag{84} \]

where \(\min \mathcal{R}_i\) denotes the smallest frame index found in the reference buffer, \(|\mathcal{R}_i|\) denotes the number of frames in the reference buffer, and \(\mathcal{R}_{\text{max}}\) stands for the maximum reference buffer occupancy. It is important to emphasize that though the next state \(\sigma_{i+1}\) depends only on \(\sigma_i\) and \(a_i\), \(a_i\) itself depends on \(a_1, ..., a_{i-1}, a_i, ..., a_n\). Formally, this can be expressed by saying that the update of the full encoder state is non-Markovian. However, some constituents of the encoder state do satisfy the Markovian property. We denote by
\[ \sigma_i^{\text{M}} = (\mathcal{R}_i, t_{i}^{\text{nd}}, t_{i}^{\text{idr}}) \tag{85} \]

the Markovian part of the state, whose update rule can be expressed as
\[ \sigma_{i+1}^{\text{M}} = \sigma(\sigma_i^{\text{M}}, f_i) \tag{86} \]

(note the dependence on \(f_i\) only). On the other hand, the remaining constituents of the encoder state
\[ \sigma_i^{\text{NM}} = (\ell_{i-}^{\text{enc}}, \ell_{i-}^{\text{raw}}, \ell_{i-}^{\text{dec}}) \tag{87} \]
are non-Markovian, since their update rule requires \( \hat{b}_i \) and \( \hat{\tau}_i \), which, in turn, depend on the entire sequence \( f \) through \( q^* \) and \( c^* \). The update rule for \( \sigma_{i}^{\text{NM}} \) is a function of the initial state \( \sigma_{1}^{\text{NM}} \) and the entire \( f \).

\[
\sigma_{i}^{\text{NM}} = \sigma(\sigma_{1}^{\text{NM}}, f).
\] (88)

9.2 Action sequence cost

Given a sequence of encoder actions \( a = (a_1, ..., a_n) \), we associate with it a sequence cost \( \rho(a) \), defined as

\[
\rho(a, \sigma) = \lambda_{\text{buf}} \sum_{i=1}^{n} \rho_{\text{buf}}(\ell_{\text{dec}}^{i}) + \lambda_{\text{dis}} \sum_{i=1}^{n} \rho_{\text{dis}}(\hat{p}_i) + \\
\lambda_{\text{bit}} \sum_{i=1}^{n} \rho_{\text{bit}}(\hat{b}_i) + \lambda_{\text{drop}} \sum_{i=1}^{n} \rho_{\text{drop}}(f_i, \hat{p}_i, \hat{p}_i^{\text{min}}) + \lambda_{\text{idr}} \sum_{i=1}^{n} \rho_{\text{idr}}(f_i, t_i, \hat{t}_{\text{idr}}) + \lambda_{\text{qp}} \sum_{i=1}^{n} \rho_{\text{qp}}(q'_i),
\] (89)

where typically \( \lambda_{\text{buf}} = 10 \), \( \lambda_{\text{dis}} = 1 \), \( \lambda_{\text{bit}} = 100 \), \( \lambda_{\text{drop}} = 0.5 \), \( \lambda_{\text{idr}} = 0.01 \), \( \lambda_{\text{qp}} = 0.1 \). The constituent terms of the cost function are defined as follows.

**Buffer cost** penalizing the estimated decoder bit buffer violation is defined as

\[
\rho_{\text{buf}}(\ell_{\text{dec}}^{i}) = h_{\text{os}}(\hat{\ell}_{\text{dec}}^{i}, \ell_{\text{dec}}^{\text{min}}) + h_{\text{os}}(\ell_{\text{dec}}^{\text{tot}} - \hat{\ell}_{\text{dec}}^{i}, \ell_{\text{dec}}^{\text{tot}} - \ell_{\text{dec}}^{\max})
\] (90)

where \( \hat{\ell}_{\text{dec}}^{i} = \hat{\ell}_{\text{dec}}^{i+} \) - \( \hat{b}_i \),

\[
h_{\text{os}}(x, y) = \frac{y}{\epsilon^2} \cdot \max \{\epsilon - \min\{x, \epsilon\}, 0\} + \max \left\{\frac{y}{\max\{x, \epsilon\}}, 1\right\} - 1
\] (91)

is a single-sided hyperbolic penalty function, and \( \epsilon \) is a small number, typically \( \epsilon \approx 10^{-6} \).

**Distortion cost** penalizing the frame distortion is given by

\[
\rho_{\text{dis}}(\hat{p}_i) = 255^2 \cdot 10^{-0.1\hat{p}_i}.
\] (92)
Drop cost  penalizing the dropped frame distortion is given by

\[
\rho_{\text{drop}}(f_i, \hat{p}_i, \hat{p}_i^{\text{min}}) = \begin{cases} 
255^2 (10^{-0.1\hat{p}_i^{\text{min}}} - 10^{-0.1\hat{p}_i}) : f_i = \text{DROP} \\
0 : \text{else}, \end{cases}
\]

(93)

where \(\hat{p}_i^{\text{min}}\) is the estimated minimum PSNR. For a dropped frame, \(\hat{p}_i^{\text{min}}\) is computed as the 0.1%-quantile of the PSNR of the difference between the original frame \(i\) and the last displayed frame \(l_i^{\text{ind}}\). For a non-dropped frame, \(\hat{p}_i^{\text{min}} = \hat{p}_i\).

IDR cost  for \(f_i = \text{IDR}\) penalizes for a too early IDR frame,

\[
\rho_{\text{idr}}(t_i, t_i^{\text{idr}}) = \begin{cases} 
\infty : t_i - t_i^{\text{idr}} < t_i^{\text{idr}}_{\text{min}} \\
0 : \text{else}. \end{cases}
\]

(94)

For \(f_i \neq \text{IDR}\), the cost is given by

\[
\rho_{\text{idr}}(t_i, t_i^{\text{idr}}) = \begin{cases} 
\infty : t_i - t_i^{\text{idr}} > t_i^{\text{gop}}_{\text{max}} \\
0 : t_i - t_i^{\text{idr}} < t_i^{\text{idr}}_{\text{min}} \\
\frac{t_i - t_i^{\text{idr}} - t_i^{\text{idr}}_{\text{min}}}{t_i^{\text{gop}}_{\text{max}} - t_i^{\text{idr}}_{\text{min}} : \text{else}}, \end{cases}
\]

(95)

penalizing for a too late IDR frame. The cost is constructed in such a way that an IDR is placed in the time interval \([t_i^{\text{idr}}_{\text{min}}, t_i^{\text{gop}}_{\text{max}}]\).

Quantizer fluctuation cost  penalizes on the deviation from the average sequence quantization parameter according to

\[
\rho_{\text{qp}}(q_i) = \max\{2q_i' - \bar{q}, 2\bar{q} - q_i'\} - 1,
\]

(96)

where \(\bar{q}'\) is the average quantization parameter in the previous non-dropped frames, computed as a decaying weighted average

\[
\bar{q}_i' = \frac{q_i'^{\text{num}}}{q_i'^{\text{den}}},
\]

(97)
where

\[ q'_{\text{num}} = \begin{cases} q'_{\text{num}} - 1 : f_i = \text{DROP} \\ \lambda q'_{\text{num}} + q_{i-1} : \text{else,} \end{cases} \]  

(98)

\[ q'_{\text{den}} = \begin{cases} q'_{\text{den}} - 1 : f_i = \text{DROP} \\ \lambda q'_{\text{den}} + 1 : \text{else,} \end{cases} \]  

(99)

The decay parameter is set \( \lambda \approx 0.99 \), and may be adjusted according to the sequence frame rate.

**Bit budget deviation cost** penalizes for action sequences resulting in a deviation from the allocated bit budget \( b_T \),

\[ \rho_{\text{bit}}(\hat{b}) = \left( \max \left\{ \frac{b_T - b_{\text{T}}}{b_T}, 0 \right\} + \max \left\{ \frac{b_{\text{T}} - b_T}{b_T}, 0 \right\} \right) \cdot h_{\text{oe}} \left( \frac{1^T \hat{b}}{b_T} \right), \]  

(100)

where

\[ h_{\text{oe}}(x) = \max \{ e^{\frac{x-1}{\epsilon}} - 1, 0 \}, \]  

(101)

and \( \epsilon = 0.1 \).

### 9.3 Action sequence optimization

Using the notion of the action sequence cost, the frame type optimization problem can be expressed as the minimization problem

\[ \min_{\mathbf{f}} \rho(\mathbf{f}, \mathbf{q}^*(\mathbf{f}), \mathbf{c}^*(\mathbf{f}), \mathbf{\hat{\sigma}}) \]  

(102)

Since the search space \( \mathcal{F}_1 \times \cdots \times \mathcal{F}_n \) is usually very large, the complexity of finding the best action sequence by exhaustive search is prohibitive. Due to the non-Markovian property of the cost function, no greedy Markov decision algorithms can be employed. However, one can construct a Markovian lower bound for \( \rho \), depending on \( f_i \) only, and allowing to prune the search space significantly.
We observe that though the estimated frame distortion $\hat{p}_i$ depends on the selection of the quantizer, it can be bounded below by the distortion achieved if $b_T$ bits were allocated to the $i$-th frame alone. The lower bound $\hat{p}_i$ on the frame distortion can be expressed as

$$\hat{p}_i = \begin{cases} \hat{p}_i & : f_i = \text{DROP} \\ \hat{p}_i(\max\{\hat{q}_i(b_T), q_{\text{min}}\}) & : \text{else,} \end{cases}$$

(103)

where $\hat{q}(b)$ is the inverse function of $\hat{b}(q)$ given by

$$\hat{q}_i(b) = \frac{\beta_i b}{b - \alpha_i b^r},$$

(104)

(the model coefficients $\alpha_i^b$ and $\beta_i^b$ depend on $f_i$). For dropped frames, the bound is exact; moreover, $\hat{p}_i^\text{min}$ can be estimated as well.

Aggregating the terms of $\rho$ that do not depend on the quantizer, the following lower bound on the action cost is obtained

$$\rho(f_i) = \lambda_{\text{dis}}\rho_{\text{dis}}(f_i, \hat{p}) + \lambda_{\text{drop}}\rho_{\text{drop}}(\hat{p}, \hat{p}_i^\text{min}) + \lambda_{\text{idr}}\rho_{\text{idr}}(f_i, t_i, t_i^{\text{idr}})$$

(105)

(though a lower bound on the buffer penalty $\rho_{\text{buf}}$ can also be devised, we do not use it here for simplicity). Note that unlike $\rho$, $\underline{\rho}$ is additive, that is

$$\underline{\rho}(f) = \sum_{i=1}^{n} \rho(a_i).$$

(106)

The lower bound $\underline{\rho}(a)$ is used in the branch and bound Algorithm 4 for solving the combinatorial minimization problem (102). The algorithm is first invoked with $f = \emptyset, \rho = \infty$, and the current state of the encoder. The order of the loop searching over all feasible frame types $\mathcal{F}_1$ should be selected to maximize the probability of decreasing the bound $\underline{\rho}$ as fast as possible. Typically, the best ordering of $\mathcal{F}$ is DROP, followed by P (if multiple reference frames are available, they are ordered by increasing display time difference relative to the current frame), followed by IDR. Though not considered here, non-IDR I frames and non-reference P or B frames can be straightforwardly allowed for as well.

In the case where complexity allocation is performed after bit allocation and the latter is not reiterated, complexity allocation may be removed from the frame type decision algorithm and performed after the
input: sequence of \( n \) frames; best frame type decision \( f \) so far; initial non-Markovian state \( \hat{\sigma}^{\text{NM}}_1 \); current Markovian state \( \sigma^M_1 \).

output: optimal action sequence \( a^* \); optimal action sequence cost \( \rho^* = \rho(a^*) \); lower bound \( \rho \).

1 if \( n \leq 0 \) then
  // Leaf reached
  2 From \( f \) and \( \hat{\sigma}^{\text{NM}}_1 \), allocate the sequence bit budget \( b_T \).
  3 Find optimal frame quantizer allocation \( q^* \).
  4 From \( f \) and \( \hat{\sigma}^{\text{NM}}_1 \), allocate the sequence encoding time budget \( \tau_T \).
  5 Find optimal frame encoding complexity allocation \( c^* \).
  6 Form action sequence \( a^* = (f, q^*, c^*) \).
  7 Compute sequence cost \( \rho^* = \rho(a^*, \hat{\sigma}) \).
  8 if \( \rho^* < \rho \) then update lower bound \( \rho = \rho^* \).
else
  9 Set \( \rho^* = \infty \).
  10 forall \( f_1 \in F_1 \) do
    11 if \( \rho(f_1) > \rho \) then continue // Prune subtree
    12 Compute the updated Markovian encoder state \( \sigma^M_2 = \sigma(\sigma^M_1, f_1) \).
    13 Add \( f_1 \) to the current \( f \) and invoke the algorithm recursively for the remaining \( n - 1 \) frames in the sequence, producing the action sequence \( a \), its cost \( \rho \), and an updated bound \( \rho \).
    14 if \( \rho < \rho^* \) then \( \rho^* = \rho \), \( a^* = a \). // Update current best action sequence
  15 end
  16 end
17 end

Algorithm 4: Branch and bound frame type decision algorithm.

optimal frame types are assigned.

10 Encoding order optimization

Given a sequence of \( n \) frames indexed by incrementing display order, the purpose of encoding order optimization is to find their optimal encoding order. For notation convenience, we denote the frame reordering as a permutation \( \pi \) of \( \{1, ..., n\} \). We also denote by \( a^*(\pi) \) and \( \rho^*(\pi) \) the optimal action sequence and its cost, respectively, found using the frame type decision algorithm from the previous section applied to the ordered sequence of frames \( \pi_1, ..., \pi_n \). Using this formalism, encoding order optimization can be expressed
as finding the permutation minimizing

$$\min_{\pi \in \Pi} \rho^*(\pi).$$  (107)

Since out-of-display-order encoding requires temporary storing the decoded frames until the next consecutive frame has been decoded, the search space of feasible permutations is constrained by the decoded picture buffer level. In H.264, the decoded picture buffer is shared with the reference buffer.

We augment the Markovian component of the encoder state with the \textit{last displayed frame index} \(l_{\text{disp}}^i\), specifying the largest frame index that has been consecutively decoded from the sequence start at the time when frame \(i\) is being encoded. The last displayed frame is updated according to

\[
l_{\text{disp}}^{i+1} = \begin{cases} l_{\text{disp}}^i : & \min\{\min R_i, \pi_i\} > l_{\text{disp}}^i + 1 \\ \max \text{con}(R_i) : & \text{else}, \end{cases}
\]

where \(\text{con}(R_i)\) denotes the largest subsequence of consecutive frame indices in \(R_i\) starting with \(\min R_i\) (e.g., if \(R_i = \{1, 2, 3, 5\}\), \(\text{con}(R_i) = \{1, 2, 3\}\); if \(R_i = \{1, 3, 4, 5\}\), \(\text{con}(R_i) = \{1\}\), etc). We also modify the state update rule for the reference buffer as

\[
R_{i+1} = \begin{cases} \overline{R}_{i+1} : & |\overline{R}_{i+1}| \leq R_{\text{max}} \\ \overline{R}_{i+1} \setminus \{k \in \text{con}(\overline{R}_{i+1}, R_{\text{max}} - |\overline{R}_{i+1}|) : k \leq l_{\text{disp}}^i\} : & \text{else}, \end{cases}
\]

where

\[
\overline{R}_{i+1} = \begin{cases} \emptyset : & f_i = \text{IDR} \\ R_i : & f_i \in \text{NONREF} \\ \{\pi_i\} \cup R_i : & f_i \in \text{REF}, \end{cases}
\]

and \(\text{con}(R_i, k)\) denotes the sequence of at most \(k\) smallest elements in \(\text{con}(R_i)\) (or less, if \(|\text{con}(R_i)| < k\)). Note that \(\pi_i\) replaces \(i\), and the update scheme now allows frames to be locked in the reference buffer until a consecutive sequence is formed.

In order to satisfy the reference buffer size constrains, \(|R_i| \leq R_{\text{max}}\) has to hold for every frame in the sequence. This condition can be verified prior to minimizing \(\rho^*(\pi)\); permutations not satisfying the reference buffer constrains are discarded from the search space. In some applications, additional constrains may apply to \(\Pi\) in order to enforce further frame ordering regularity. For example \(\Pi\) may be restricted to
input: sequence of $n$ frames.
output: optimal permutation $\pi$; optimal action sequence $a^*$; optimal action sequence cost $\rho^* = \rho(a^*)$.

initialization: Set $\rho = \rho^* = \infty$.

forall $\pi$ do
1 Verify reference buffer conditions.
2 if $|R_i| > R_{max}$ for any $i$ then continue
3 Invoke the frame type decision algorithm on the sequence of frames $\pi_1, ..., \pi_n$, producing the action sequence $a$, its cost $\rho$, and an updated bound $\rho$.
4 if $\rho < \rho^*$ then $\rho^* = \rho$, $a^* = a$, $\pi^* = \pi$.
6 end

Algorithm 5: Encoding order optimization algorithm.

few pre-defined ordering patterns providing regular single- or multi-level temporal scalability of the encoded sequence. The encoding order optimization procedure is summarized in Algorithm 5.

References

Figure 2: Example of buffer models.
Figure 3: Sequence-level controller flow diagram.
Figure 4: Cost functions used in the controller.
Figure 5: Cost functions used in the controller (cont).