Designing Autonomous Layered Video Coders

Nicholas Mastronarde and Mihaela van der Schaar

ABSTRACT

Layered compression paradigms such as scalable, multiple description, and multi-view video coding, require coupled encoding decisions among layers in order to achieve optimal distortion performance under buffer constraints. Moreover, due to the dynamic and time-varying source characteristics, and temporal coupling of encoding decisions through the buffer constraints, it is not only necessary to consider the immediate rate-distortion impact of encoding decisions, but also their long-term rate-distortion impact. In other words, optimal encoding decisions must consider the coupling between layers and the coupling across time. In many scenarios, however, it may be impractical to make joint coding decisions for all of the layers. For instance, a two layer bitstream may be coded using different encoders for the base layer and enhancement layer, each with its own autonomous control plane; or, if the same encoder is used for multiple layers, then a joint decision process, which considers the aforementioned dependencies, may be too complex. In this paper, we propose a framework for autonomous decision making in layered video coders, which decouples the decision making processes at the various layers using a novel layered Markov decision process. We illustrate how this framework can be applied to decompose the decision processes for several typical layered video coders with different dependency structures and we observe that the performance of the proposed decomposition highly depends on the ability of the layers to model each other.

Keywords: Foresighted rate control, Layered Markov decision process, Autonomous layered video coding.

I. INTRODUCTION

Layered video coders compress video sources into multiple bitstreams, which can be selectively transmitted depending on the available network bandwidth [5] [15]. These bitstreams can be encoded to adhere to independent or shared buffer constraints using a video buffer verifier [1] [2]. In order to achieve optimal distortion performance under these buffer constraints, decisions on how to encode each layer must be made (i) in a *foresighted* manner, because of the temporal coupling of encoding decisions through the buffer constraints, and (ii) *jointly*, because of the dependencies among the layers .

Foresighted optimizations not only consider the immediate rate-distortion impact of encoding decisions,

but also their long-term impact. This is important, because the number of bits allocated to encode a data unit (e.g. video picture) at the current time determines the amount of bits available to encode future data units. Hence, foresighted encoding decisions can significantly improve the long-term rate-distortion performance of encoders.

In this paper, we formulate the buffer control problem as a Markov decision processes (MDP). Using MDP formalisms, foresighted decisions can be made by either doing multi-pass encoding [11][12], to learn the rate-distortion characteristics of the video sequence before making final encoding decisions, or they can be learned at run-time using online learning techniques [13][14]. Importantly, it has been shown that video sequence characteristics and video traffic can be accurately modeled as a Markov process [4]. Hence, MDPs provide a rigorous and formal methodology for using these existing models to improve video encoder performance.

Dependencies in layered video coders come in two flavors: *data dependencies* and (rate) *budget dependencies*. Data dependencies make it so that an enhancement layer can only be encoded or decoded after the base layer is encoded or decoded. For example, additional B frames in a temporal enhancement layer can only be encoded or decoded after the low frame-rate I and P frames in the base layer are encoded or decoded. Importantly, data dependencies also imply that the distortion and encoded rate of the enhancement layer depend on how the base layer has been encoded. Meanwhile, budget dependencies govern the allocation of bits to the various layers. The budget dependencies are not important if the layers have *independent* rate budgets; however, they become crucial if the base layer and enhancement layers *share* a rate budget because this forces the coders to make tradeoffs between the quality of the base layer and the enhancement layer(s), the quality of the various descriptions, or the quality of views from different cameras.

Despite the need for joint encoding decisions to cope with the data and budget dependencies, and to achieve optimal long-term distortion performance under buffer constraints, there are a number of reasons that it may be impractical or even impossible to make joint encoding decisions across all of the layers in some scenarios:

1. ***Computational Complexity:*** A centralized foresighted optimization can be orders of magnitude more complex than a distributed foresighted optimization.
2. ***Design and Implementation Time:*** If a layered video encoder is constructed from two or more legacy and/or proprietary encoders, each with their own existing autonomous controllers, then additional

software would have to be designed to jointly control all of the layers' decisions in a centralized way. In this case, significantly more engineering effort is required if the layers must act jointly.

3. ***Distributed Encoders:*** In emerging multi-view video capture systems, cameras and their associated buffers are not always collocated, and inter-camera communication must be minimized [16]; hence, a centralized controller is impractical in this scenario, because it would require every camera to frequently communicate with every other camera or with some centralized entity.
4. ***Differential Information:*** For multiple description encoders, encoding decisions for different descriptions with their own independent buffers may be conditioned on the dynamic bandwidth constraints of different network paths [18]. In this scenario, there may only be marginal benefit (or none at all) for using a centralized controller.

If a centralized control plane like the one in Fig. 1(a) is impractical to implement for any of the above reasons, then a distributed control plane like the one in Fig. 1(b) can be deployed, which allows each layer to autonomously select its own encoder settings based on limited or no information from the other layers.

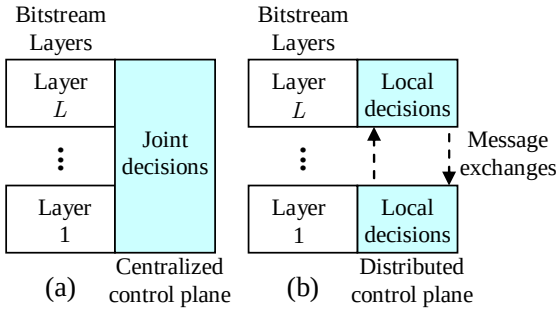


Fig. 1. (a) Centralized control plane; (b) Distributed control plane (message exchanges are optional).

In this paper, we introduce a framework for autonomous decision making in layered video coders, which takes into account the data and budget dependencies among the video layers. To illustrate how different video applications with different dependencies can be implemented in our framework, and how these dependencies impact the decision processes, we consider a variety of layered video coders, including scalable [15], multiple description [5], and multi-view [8] encoders. Importantly, the proposed framework and methodologies are general enough to be applied in various buffer-constrained encoder settings such as video buffer verifiers [1][2] and video complexity verifiers [3]; however, for simplicity, we will focus exclusively on video buffer verifiers in this paper.

Our contributions are as follows:

- We formulate the layered rate control problem as a centralized MDP, which not only considers the

immediate rate-distortion impact of encoding decisions on the current frame, but also their long-term impact on future frames, thereby optimizing the long-term rate-distortion performance of the encoder.

- We propose a methodology for decomposing the centralized MDP into a novel layered MDP, which enables each layer to autonomously select its own encoding actions. We analytically define the *performance gap*, which measures the performance loss associated with a layered MDP compared to the optimal centralized MDP.
- We apply our decomposition methodology to three illustrative examples with different data and budget dependency structures. We identify three *decomposition principles*, which we use to formulate decomposable approximations of centralized MDPs in our illustrative examples.
- We discuss how the proposed framework can be applied to emerging Reconfigurable Video Coding MPEG standards and how it can be extended to include online adaptation to dynamically changing source and traffic characteristics or network constraints.

Importantly, this paper is not intended to show all possible combinations of data and budget dependencies that can occur in layered encoder settings. Instead, examples are selected only to illustrate the proposed methodology when dealing with various data and budget dependencies that may be encountered under different application scenarios. In other words, some new settings may benefit from the illustrative decomposition examples in this paper, while others may require new or refined solution methodologies.

The paper is organized as follows. In Section II, we present our general data dependency model for layered video coders. In Section III, we present three budget dependency models. In Section IV, we present the centralized Markov decision process based problem formulation and informally outline the proposed decomposition methodology. In Section V, we present three detailed decomposition case studies to illustrate the proposed decomposition methodology. In Section VI, we discuss how the proposed framework can be applied to emerging Reconfigurable Video Coding MPEG standards and how to integrate dynamic online adaptation into the framework. In Section VII, we present our experimental results and we conclude the paper in Section VIII.

II. Data Dependency Models

In this section, we propose a general model for describing the inter-layer data dependencies arising in layered video coders. We illustrate three typical instantiations of the model: one based on a scalable coder, one

based on a multiple description coder, and one based on a multi-view encoder. These examples are designed to illustrate several representative data dependency structures that are typical in layered video coders, and are by no means intended to exhaustively explore all possible dependency structures and encoder instantiations.

We assume that the layered video encoder generates a set of L video layers[1] denoted by $\mathcal{L} = \{1,\ldots,L\}$. In general, the data dependencies among layers can be expressed using a directed acyclic graph (DAG), where each node of the graph represents one layer and each edge of the graph directed from layer l' to layer l represents the dependence of layer l on l'. Formally, we define a partial relationship between two layers by writing $l' \prec l$, if layer l' is an *ancestor* of layer l. In other words, $l' \prec l$ means that layer l cannot be encoded or decoded unless layer l' has been encoded or decoded.

Example: *Inter-layer data dependencies:* Consider a temporal scalable bitstream comprising a low frame-rate base layer (say layer l') containing only I and P frames and an enhancement layer (say layer l) containing only B frames. The B frames cannot be encoded or decoded unless the I and P frames are encoded or decoded (i.e. $l' \prec l$).

We define the lth layer's set of ancestors[2] as

$$\Gamma_l = \{l' \in \mathcal{L} \mid l' \prec l\}.$$

We note that if $\Gamma_l = \varnothing$ then layer l is an *independently encodable or decodable layer*. On the other hand, if $\Gamma_l \neq \varnothing$, then there exists an $l' \in \Gamma_l$ such that $\Gamma_{l'} = \varnothing$. In other words, every chain of ancestors terminates at atleast one independently encodable or decodable layer (which is usually the base layer for a scalable coder or any side description for a multiple description coder). We further define the set

$$\Lambda_l = \{l\} \cup \Gamma_l,$$

which represents an independently encodable or decodable set of layers that contains layer l and all of its ancestors. In other words, if the set of layers Λ_l is available at a receiver, then it will be able to decode layer l and its ancestors.

We model each layer $l \in \mathcal{L}$ as a sequence of *data units* (DUs) indexed by n. In this paper, we assume that a DU corresponds to one video picture. We interchangeably refer to a DU that belongs to layer l as the *lth layer's DU* or a *layer-l DU*.

At every DU index n, we allow each layer-l DU (for all $l \in \mathcal{L}$) to be coded under different encoder configurations. We denote the set of configurations available to encode layer-l DUs as $\mathcal{A}_l$, and let $a_l \in \mathcal{A}_l$

[1] Throughout this paper, we will use the terms "layers, "descriptions," and "views" interchangeably.

[2] In typical scalable video coding nomenclature, Γ_l is the set of sublayers that must be decoded to decode layer l.

represent one such configuration. We refer to a_l as the lth layer's *action* and $\mathcal{A}_l$ as its *action set*. Although actions are selected per DU (i.e. a new action can be chosen at every DU index n), we omit this from the notation until Section III.

Example: *Encoder actions:* Encoder actions include any decisions that are made at encoding time that impact the rate-distortion performance. For instance, the choice of sub-pixel motion accuracy, macroblock partition size, quantizer parameter (QP), deblocking filter, entropy coding scheme, motion estimation algorithm, frame type etc. are all possible encoder actions. In the illustrative results in this paper, we limit the actions to the choice of QP.

Let us assume that the joint encoding action $\boldsymbol{a} = (a_1,\ldots,a_L) \in \boldsymbol{\mathcal{A}}=\mathcal{A}_1 \times \cdots \times \mathcal{A}_L$ is used to encode the L layers, where action a_l is used to encode layer l. The encoded rate (bits/DU) and decoded distortion (mean squared error) of the set of layers $\Lambda_l = \{l\} \cup \Gamma_l$ not only depend on the action a_l, but also on the actions $a_{l'} \in \boldsymbol{a}(\Gamma_l) = \left\{ [\boldsymbol{a}]_{l'} \mid l' \in \Gamma_l \right\}$ of the lth layer's ancestors, where $[\boldsymbol{a}]_{l'} = a_{l'}$ is the l'th component of the joint action $\boldsymbol{a} = (a_1,\ldots,a_L)$. This dependency exists because the actions $a_{l'} \in \boldsymbol{a}(\Gamma_l)$ shape the reference DUs that layer l depends on. We denote these reference DUs as $f_l(\boldsymbol{a}(\Gamma_l))$.

Example: *Reference DUs:* Consider the triplet of frames I-P-B. The B frame in the temporal enhancement layer (say layer l) is bi-directionally predicted from the decoded I and P frames in the base layer (say layer l'); therefore, in this example, $f_l(\boldsymbol{a}(\Gamma_l))$ represents the decoded I and P frames that are used as references for predicting the B frame.

Encoding the DU composed of the set of layers $\Lambda_l = \{l\} \cup \Gamma_l$ yields the decoded distortion $d_{\Lambda_l}(a_l, \boldsymbol{a}(\Gamma_l))$ and encoded rate $b_{\Lambda_l}(a_l, \boldsymbol{a}(\Gamma_l))$. As we will see in the following three subsections, the precise definition of these quantities depends on the application and its dependency structure.

It is important to note that, in this paper, we only consider *inter-layer* data dependencies, which dictate how one layer's actions impact another layer's rate-distortion behavior; consequently, we ignore *intra-layer* data dependencies, which dictate how actions at DU index n impact the rate-distortion behavior of future DUs, at indices $n+1, n+2, \ldots$, in the same layer.

Example: *Intra-layer data dependencies:* The QP selected for an I frame impacts the rate-distortion behavior of all subsequent DUs in the same group of pictures.

Although we ignore intra-layer data dependencies, encoding decisions at DUs $n+1, n+2, \ldots$ are still coupled with the encoding decision for DU n through the post-encoding buffer (see Section III). For

illustration of how intra-layer dependencies can be dealt with, we refer the interested reader to [6] [19].

A. Scalable Coding Example

For a scalable video coder, we let layer $l = 1$ represent the *base layer* and layers $l \in \{2,\ldots,L\}$ represent the $L-1$ *enhancement layers* (e.g. temporal, spatial, or signal-to-noise ratio enhancements). We assume that each layer can be generated by different video standards (e.g. the base layer can be generated by an MPEG-2 based coder while the enhancement layers can be generated by an H.264/AVC based coder). Fig. 2 illustrates a simple example of temporal scalability with $L = 2$ layers. Note how we index DUs and how we define a "2-layer DU" as the combination of a "Layer-1 DU" and a "Layer-2 DU." Spatial and signal-to-noise ratio enhancements can be represented similarly.

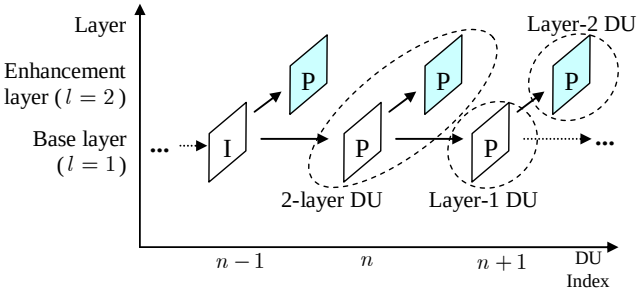


Fig. 2. Scalable video coding example: Temporal scalability with one base layer and one enhancement layer.

Due to the hierarchical relationship between layers, the lth layer can only be encoded after layers $l' \in \{1,\ldots,l-1\}$ have been encoded; hence, $\Gamma_l = \{1,\ldots,l-1\}$, $\Lambda_l = \{1,\ldots,l\}$, and $\boldsymbol{a}(\Gamma_l) = \{a_1,\ldots,a_{l-1}\}$. The base layer's action a_1 determines the base layer's distortion $d_1(a_1)$ and encoded rate $b_1(a_1)$. We denote the resulting decoded base layer, which is used as a reference by layer 2, as $f_2(a_1)$, where $\{a_1\} = \boldsymbol{a}(\Gamma_2)$. Given $f_2(a_1)$, the first enhancement layer's action a_2 determines its distortion reduction[3] $\Delta d_2(a_2, f_2(a_1))$ and encoded rate $b_2(a_2, f_2(a_1))$. We denote the resulting references used by layer 3 as $f_3(a_1,a_2)$, where $\{a_1,a_2\} = \boldsymbol{a}(\Gamma_3)$.

In general, given the references generated by layers $l' \in \Gamma_l$, i.e. $f_l(a_1,\ldots,a_{l-1})$, the lth layer's action a_l, for $l \in \{2,\ldots,L\}$, determines its distortion reduction $\Delta d_l(a_l, f_l(a_1,\ldots,a_{l-1}))$ and encoded rate $b_l(a_l, f_l(a_1,\ldots,a_{l-1}))$ as illustrated in Fig. 3(a). When we discuss the proposed decomposition methodology in Section IV and the decompositions in Section V, it will be important to understand why $\Delta d_l(a_l, f_l(a_1,\ldots,a_{l-1}))$ and $b_l(a_l, f_l(a_1,\ldots,a_{l-1}))$ are functions of the reference frames $f_l(a_1,\ldots,a_{l-1})$ and not

[3] The distortion reduction at layer 2 represents the decrease in distortion relative to the base layer's distortion (i.e. layer 1). In other words, decoding the first enhancement layer in addition to the base layer yields total distortion $d_{\Lambda_2}(a_1,a_2) = d_1(a_1) - \Delta d_2(a_2, f_2(a_1))$.

the actions $a_1,\ldots,a_{l-1}$. To understand this, consider the following example.

Example: *Impact of* $f_l(a_1,\ldots,a_{l-1})$*:* Consider again the I-P-B frame triplet. As before, the encoding actions selected for the I and P frames in the base layer determine the reference information used to predict the B frame in the enhancement layer. Regardless of the actions used to encode the I and P frames, if they result in an identical set of reference pixels, then the B frame's distortion reduction and encoded rate only depend on those reference pixels and the enhancement layer's action.

The l-layer DU comprising layers $\Lambda_l = \{1,\ldots,l\}$ has distortion[4]

$$d_{\Lambda_l}(a_1,\ldots,a_l) = d_1(a_1) - \sum_{k=2}^{l} \Delta d_k(a_k, f_k(a_1,\ldots,a_{k-1})), \tag{1}$$

and encoded rate

$$b_{\Lambda_l}(a_1,\ldots,a_l) = b_1(a_1) + \sum_{k=2}^{l} b_k(a_k, f_k(a_1,\ldots,a_{k-1})). \tag{2}$$

In words, decoding more layers additively decreases the video distortion at the expense of an additively increased bit-rate. Fig. 3(a) illustrates the relationships among the quantities on the right hand sides of (1) and (2). Fig. 3(b) illustrates the construction of the l-layer DU and its parameters on the left hand sides of (1) and (2).

When we discuss the budget dependency models in Section III, it will be important to understand why $\Delta d_l(a_l, f_l(a_1,\ldots,a_{l-1}))$ and $b_l(a_l, f_l(a_1,\ldots,a_{l-1}))$ are non-deterministic. To understand this, consider the following example.

Example: *Non-deterministic rate-distortion behavior:* Consider again the I-P-B frame triplet. The actual rate-distortion behavior of the B frame depends on how well it is correlated with its reference frames (i.e. the decoded I and P frames). For instance, if the B frame is the same as the I or P frame, then $b_l(a_l, f_l(a_1,\ldots,a_{l-1}))$ will be quite small; however, if there is a lot of motion in the sequence, then the B frame will not be well correlated with either of the reference frames. If this correlation is not known, then $\Delta d_l(a_l, f_l(a_1,\ldots,a_{l-1}))$ and $b_l(a_l, f_l(a_1,\ldots,a_{l-1}))$ are non-deterministic and depend on the video source's characteristics.

[4] This distortion model assumes that there is only one independently decodable ancestor layer (i.e. the base layer). If the scalable video encoder has more than one base layer, then a different model is needed.

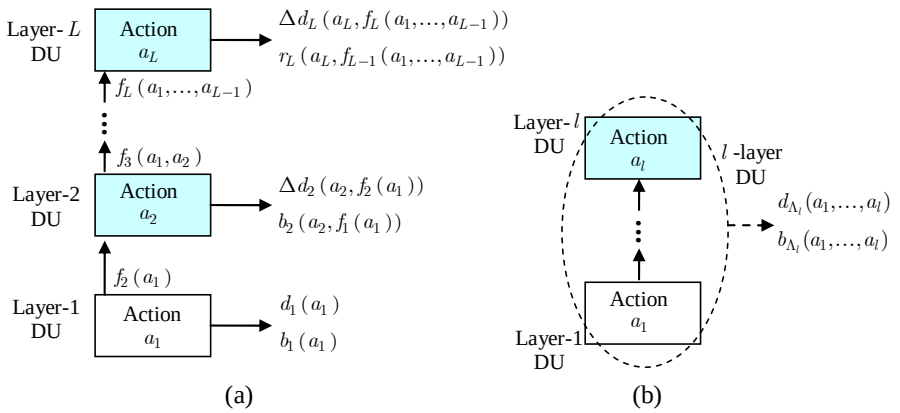


Fig. 3. Scalable coding data dependencies: (a) Relationships among the layer-l DUs and their parameters; (b) Construction of the l -layer DU, and its parameters.

B. Multiple Description Coding Example

We consider an illustrative multiple description coder with $L = 2$ descriptions generated by temporal or spatial information splitting, syntactic-semantic splitting, or layered coding [5]. Fig. 4(a) illustrates a simple example of multiple description coding with temporal splitting. The dotted oval indicates that the two DUs (i.e. the P frames in description 1 and description 2) have the same DU index.

Due to the asymmetric relationship between the two descriptions, each is independently encodable and decodable (i.e. there are no data dependencies between the descriptions). In our framework, a multiple description coder with two descriptions can be modeled as having $L = 3$ layers, where layer $l = 1$ represents the first description, layer $l = 2$ represents the second description, and layer $l = 3$ represents a *virtual* layer comprising both descriptions. Hence, $\Gamma_1 = \varnothing$, $\Gamma_2 = \varnothing$, and $\Gamma_3 = \{1,2\}$; and, $\Lambda_1 = \{1\}$, $\Lambda_2 = \{2\}$, and $\Lambda_3 = \{1,2,3\}$. The first description's action a_1 determines its distortion $d_1(a_1)$ and encoded rate $b_1(a_1)$. Similarly, the second description's action a_2 determines its distortion $d_2(a_2)$ and encoded rate $b_2(a_2)$. Note that layer $l = 3$ (i.e. the virtual layer made of description 1 and description 2) does not have its own action, i.e. $\mathcal{A}_3 = \varnothing$, because it is merely the combination of the first two layers. We write $a_3 = \mathbf{0}$ to indicate a null action and $d_3(\mathbf{0}) = 0$, $b_3(\mathbf{0}) = 0$ to indicate the corresponding decoded distortion and encoded rate, respectively. If both descriptions are received, then the final decoded distortion becomes $d_{\Lambda_3}(a_1,a_2)$, which is less than both $d_1(a_1)$ and $d_2(a_2)$ [5], and the final encoded rate becomes $b_{\Lambda_3}(a_1,a_2) = b_1(a_1) + b_2(a_2)$.

Fig. 4(b) illustrates the dependencies among the multiple description coders' layers and the corresponding parameters. As we mentioned in Section II.A, we do not explicitly consider the intra-layer dependencies, but we account for the temporal coupling of decisions at different DU indices through the post-encoding buffer (see Section III).

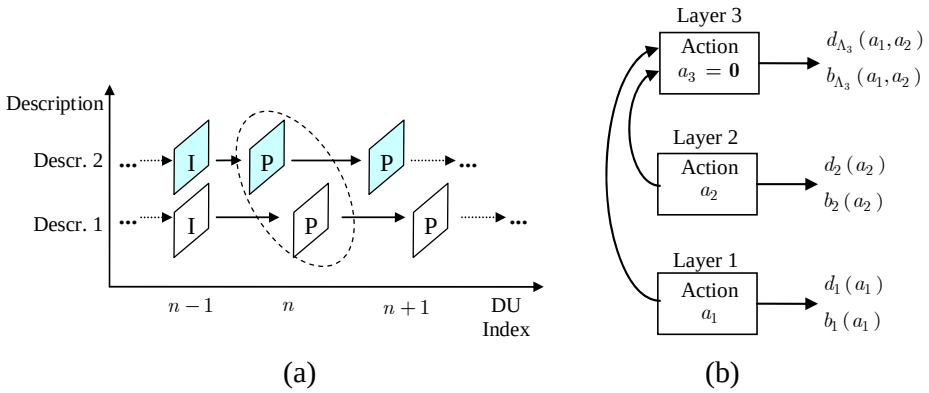


Fig. 4. Multiple description coding example: (a) Temporal splitting with two descriptions. (b) Multiple description coding data dependencies.

C. Multi-view Coding Example

For illustration, we consider a very simplistic multi-view coding example with L independently encodable or decodable views from L cameras (i.e. there are no data dependencies among the L views). Hence, $\Gamma_l = \varnothing$ and $\Lambda_l = \{l\}$. The lth sequence's action a_l determines its distortion $d_l(a_l)$ and encoded rate $b_l(a_l)$. Although we consider a very simple example here, our data dependency framework can be applied to multi-view coders with various inter-view dependencies such as those in [7] and [8].

III. BUDGET DEPENDENCY MODELS

In the previous section, we discussed how inter-layer data dependencies couple the descendent layers' rate-distortion performance with the actions of their ancestor layers. In this section, we discuss budget dependency models, which define how the encoding action taken for the current DU in one or more layers impacts the available bit budget for future DUs in one or more layers. By combining different data dependency models and different budget dependency models, a large number of application's and constraints can be represented.

In this paper, we use the budget models within a buffered rate-control framework [1] in order to regulate the encoded rate of one or more layers; however, a similar approach can be applied in a video complexity verifier setting as well [3], by substituting encoded rate for decoding complexity. Importantly, the budget dependency models proposed in this section are just for illustration. Other budget dependency structures may be required for other application scenarios.

Fig. 5 illustrates the three considered buffer models. In Fig. 5, an arrow pointing from a layer-l DU to a buffer indicates that b_l bits are deposited into the buffer after the DU is encoded using action a_l. The arrows from the buffers back to the DUs indicate that the action selected for encoding each DU is conditioned on the

current state of one or more of the buffers.

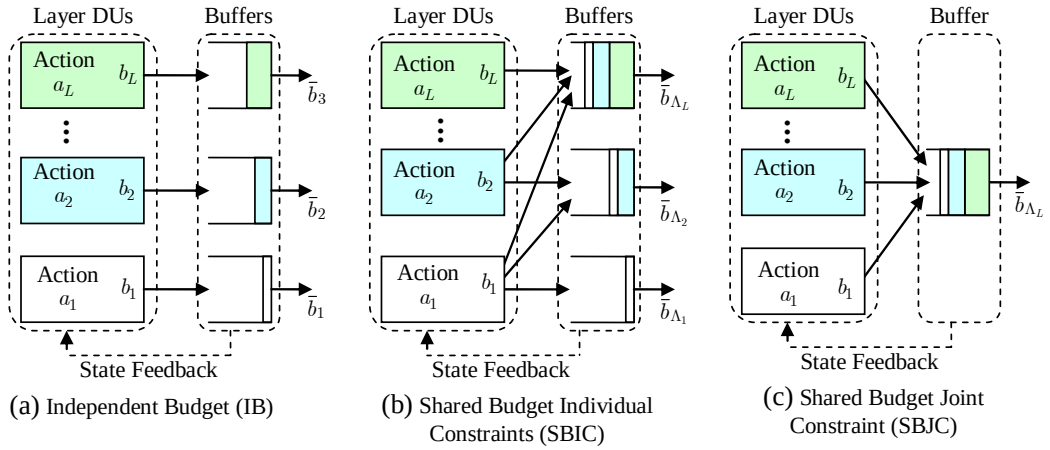


Fig. 5. Illustrative budget dependency models: (a) Independent Budget model (IB); (b) Shared Budget Individual Constraints model (SBIC); (c) Shared Budget Joint Constraint model (SBJC).

A. Independent Budget (IB)

The *independent budget* (IB) buffer model illustrated in Fig. 5(a) assigns to each layer one buffer, which independently regulates the encoded rate of that layer's bitstream. We assume that the buffer at layer l drains at the rate $\bar{b}_l$.

Under the IB model, we denote the lth layer's buffer's occupancy prior to coding the nth DU as q_l^n, where the subscript indicates that the lth layer's buffer only regulates the rate of layer l. The buffer's occupancy evolves recursively from DU index n to DU index $n+1$ as follows [1] [2]:

$$\begin{aligned} q_l^{n+1} &= \min\left\{\left[q_l^n + b_l^n\left(a_l^n, f_l^n\left(\boldsymbol{a}^n\left(\Gamma_l\right)\right)\right) - \bar{b}_l\right]^+, q_l^{\max}\right\} \\ q_l^0 &= q_l^{\text{init}}, \end{aligned} \tag{3}$$

where q_l^{init} is the initial buffer occupancy; $q_l^{\max}$ is the maximum allowable buffer occupancy; $[Y]^+ = \max\{Y, 0\}$; $\bar{b}_l$ is the buffer's drain rate; $\boldsymbol{a}^n\left(\Gamma_l\right)$ is the set of the lth layer's ancestor's actions; and, $b_l^n\left(a_l^n, f_l^n\left(\boldsymbol{a}^n\left(\Gamma_l\right)\right)\right)$ is the number of bits allocated to the nth layer-l DU given the lth layer's action a_l^n and its reference frames $f_l^n\left(\boldsymbol{a}^n\left(\Gamma_l\right)\right)$.

The *buffer state* at layer l represents the buffer's occupancy $q_l \in [0, q_l^{\max}]$; however, since q_l is continuous, we need to discretize it in order to formulate the buffered rate-control problem as a discrete-time Markov decision process in Section IV. To this end, we assume that a function ψ_l maps the buffer occupancy into one of η_l discrete values, i.e.

$$\psi_l\left(q_l\right) = s_l \in \left\{s_l^1, \ldots, s_l^{\eta_l}\right\}.$$

We refer to these values as *buffer state descriptions*. Accordingly, the set of buffer states at layer l is finite

and can be expressed as

$$\mathcal{S}_l = \{ s_l \mid s_l = \psi_l(q_l),\ q_l \in [0, q_l^{\max}] \}.$$

Due to the fact that $b_l^n(a_l^n, f_l^n(\boldsymbol{a}^n(\Gamma_l)))$ is non-deterministic (see Section II.A), the state transition for each layer's buffer is also nondeterministic. The l th buffer's transition (defined recursively in (3)) can be modeled as a controllable Markov process with transition probabilities

$$\{ p(s_l' \mid s_l, a_l, f_l(\boldsymbol{a}(\Gamma_l))) : s_l' \in \mathcal{S}_l \}, \tag{4}$$

where $s_l = s_l^n$ and $s_l' = s_l^{n+1}$ are the l th layer's buffer states at the beginning of DU indices n and $n+1$, respectively. Thus, the transition of the joint buffer state $\boldsymbol{s} = (s_1, \ldots, s_L) \in \boldsymbol{\mathcal{S}} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_L$ given the joint action $\boldsymbol{a} = (a_1, \ldots, a_L) \in \boldsymbol{\mathcal{A}} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_L$ can be written as

$$p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}) = \prod_{l=1}^{L} p(s_l' \mid s_l, a_l, f_l(\boldsymbol{a}(\Gamma_l))). \tag{5}$$

This expression intuitively follows from Fig. 5(a), which illustrates that the l th layer's buffer transition is independent of the other layers' buffer transitions given $f_l(\boldsymbol{a}(\Gamma_l))$ (i.e. the number of bits arriving in the base layer's buffer neither depends on, nor impacts, the number of bits arriving in the enhancement layer's buffer if the enhancement layer knows $f_l(\boldsymbol{a}(\Gamma_l))$).

The uncertainty in the value of $b_l^n(a_l^n, f_l^n(\boldsymbol{a}^n(\Gamma_l)))$ (and therefore the uncertainty of the next state) can be reduced by modeling the video sequence characteristics and rate traffic as a Markov process, with *video source states* representing different frame types and activity levels [4]. In this paper, to emphasize the proposed decomposition methodology, we do not directly model the source characteristics in this manner. However, such models can be easily integrated into our framework to improve performance at the expense of increased memory and computational complexity.

When layer l takes action a_l in state s_l, it incurs a *buffer cost* $g_l(s_l, a_l, f_l(\boldsymbol{a}(\Gamma_l)))$, which penalizes the layer as its buffer fills, thereby protecting against overflows that may result from a sudden burst in buffer arrivals. We present illustrative buffer cost models in Appendix B.

B. Shared Budget Individual Constraints (SBIC)

The *shared budget individual constraints* (SBIC) buffer model illustrated in Fig. 5(b) uses the buffer at layer l to regulate the encoded rate of the layers in the set $\Lambda_l = \{l\} \cup \Gamma_l$[5], instead of only regulating layer l as in the IB model. In this way, each individual bitstream Λ_l, for all $l \in \boldsymbol{\mathcal{L}}$, is forced to adhere to a maximum

[5] It is important to note that, under the SBIC model, the budget dependency structure depends on the application specific data dependency structure (i.e. $\Lambda_l = \{l\} \cup \Gamma_l$, for all $l \in \boldsymbol{\mathcal{L}}$); however, it is also possible to have a buffer model where layers have no data dependencies, but still have shared buffers.

budget (as in the IB buffer model), but has the flexibility to share its budget among its various layers (unlike the IB buffer model). In appendix A, we describe the conditions under which this extra flexibility can improve the overall rate-distortion performance of the layered video coder.

Under the SBIC model, we assume that the buffer at layer l drains at the rate $\bar{b}_{\Lambda_l}$, where $\bar{b}_{\Lambda_l}$ is the maximum average channel rate for the bitstream composed of the layers in the set Λ_l. We denote the lth layer's buffer's occupancy prior to encoding the nth DU as $q^n_{\Lambda_l}$, where the subscript indicates that the lth layer's buffer regulates the combined rate of all layers in Λ_l. The lth layer's buffer evolves recursively from DU index n to DU index $n+1$ as follows

$$\begin{aligned} q^{n+1}_{\Lambda_l} &= \min\left\{\left[q^n_{\Lambda_l} + b^n_{\Lambda_l}(a^n_l, \boldsymbol{a}^n(\Gamma_l)) - \bar{b}_{\Lambda_l}\right]^+, q^{\max}_{\Lambda_l}\right\} \\ q^0_{\Lambda_l} &= q^{\text{init}}_{\Lambda_l}, \end{aligned} \tag{6}$$

where $q^{\text{init}}_{\Lambda_l}$ is the initial buffer occupancy; $q^{\max}_{\Lambda_l}$ is the maximum allowable buffer occupancy; $\bar{b}_{\Lambda_l}$ is the buffer's drain rate; and $b^n_{\Lambda_l}(a^n_l, \boldsymbol{a}^n(\Gamma_l))$ is the number of bits allocated to the nth DU in the bitstream Λ_l. Note that we use $\boldsymbol{a}^n(\Gamma_l)$ here instead of $f(\boldsymbol{a}^n(\Gamma_l))$, because $b^n_{\Lambda_l}(a^n_l, \boldsymbol{a}^n(\Gamma_l))$ depends directly on the actions (see (2)).

As in the IB buffer model, a state at layer l represents the quantized buffer occupancy; hence, the lth layer's buffer state set is $\mathcal{S}_l = \left\{s_l \mid s_l = \psi_{\Lambda_l}\left(q_{\Lambda_l}\right),\ q_{\Lambda_l} \in [0, q^{\max}_{\Lambda_l}]\right\}$.

Due to the uncertainty in $b^n_{\Lambda_l}(a^n_l, \boldsymbol{a}^n(\Gamma_l))$, each buffer's state transition is non-deterministic. The state transition at video layer l (defined recursively in (6)) can be modeled as a controllable Markov process with transition probabilities

$$\left\{p\left(s'_l \mid s_l, a_l, \boldsymbol{a}(\Gamma_l)\right) : s'_l \in \mathcal{S}_l\right\}. \tag{7}$$

Thus, the transition of the joint buffer state $\boldsymbol{s}$ given the joint action $\boldsymbol{a}$ can be written as

$$p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}) = \prod_{l=1}^{L} p\left(s'_l \mid s_l, a_l, \boldsymbol{a}(\Gamma_l)\right), \tag{8}$$

where the lth layer's buffer transition is independent of the other layer's buffer transitions given $\boldsymbol{a}(\Gamma_l)$. The form of (8) intuitively follows from Fig. 5(b), which shows that the actions of the layers in Λ_l determine the number of bits that enter the lth buffer.

If layer l is in state s_l, layer l takes action a_l, and layers $l' \in \Gamma_l$ take actions $\boldsymbol{a}(\Gamma_l)$, then layer l incurs a buffer cost $g_{\Lambda_l}(s_l, a_l, \boldsymbol{a}(\Gamma_l))$. We present illustrative buffer cost models in Appendix B.

C. Shared Budget Joint Constraints (SBJC)

The *shared budget joint constraint* (SBJC) buffer model illustrated in Fig. 5(c) is a special case of the

SBIC buffer model. The SBIC buffer model reduces to the SBJC model when the buffer sizes and drain rates of layers $l \in \{1,\ldots,L-1\}$ are infinite and their buffer costs are zero. In other words, the SBJC buffer model regulates the encoded rate of the entire bitstream Λ_L[6], without imposing any constraints on individual sublayers or side descriptions.

IV. Decomposition Methodology

In this subsection, we begin by introducing the centralized problem formulation based on a conventional Markov decision process (MDP). Subsequently, we informally outline the decomposition methodology proposed in this paper.

A. Centralized Problem Formulation

We formulate the buffered rate-control problem for layered video coders as an MDP. MDPs enable *foresighted* decisions, which not only consider the immediate rate-distortion impact of coding decisions, but also their long-term rate-distortion impact. This is important in our setting, because the number of bits allocated to encode a DU at the current time determines the amount of bits available to encode future DUs. Hence, foresighted encoding decisions can significantly improve the long-term rate-distortion performance of encoders, for a given buffer model.

Formally, an MDP is a tuple $\langle \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{A}}, J, p, \gamma \rangle$, where $\boldsymbol{\mathcal{S}}$ is a set of states; $\boldsymbol{\mathcal{A}}$ is a set of actions; $J : \boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{A}} \mapsto \mathbb{R}$ is a cost function; $p : \boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{A}} \times \boldsymbol{\mathcal{S}} \mapsto [0,1]$ is a transition probability function; and $\gamma \in [0,1)$ is a discount factor. In the proposed framework, the set of actions is $\boldsymbol{\mathcal{A}} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_L$, where $\mathcal{A}_l$ is the lth layer's set of encoding parameter configurations as described in Section II; and, the set of states and the transition probability function depend on the budget dependency model as described in Section III. We will define cost functions for several illustrative examples in Section V.

We define a *Markov decision policy* π as a mapping from joint states to joint actions, i.e. $\pi : \boldsymbol{\mathcal{S}} \mapsto \boldsymbol{\mathcal{A}}$. In our setting, the policy π simply dictates which encoding actions should be taken by each layer given the current buffer states. The goal of the MDP is to find the optimal stationary Markov policy π^*, which minimizes *the expected sum of discounted costs*

$$E_\pi \left(\sum_{n=0}^{\infty} (\gamma)^n \, J^n \mid \boldsymbol{s}^0 \right) \tag{9}$$

where the parameter $\gamma \in [0,1)$ is the discount factor, which defines the relative importance of present and

[6] If there are no data dependencies, then the SBJC model can regulate the encoded rate of all layers in $\boldsymbol{\mathcal{L}}$.

future rewards[7]; $\boldsymbol{s}^0$ is an initial starting state; and, J^n is the cost incurred when coding the n th DU index (see Section V for several illustrative cost function definitions).

We will find it useful to express the expected sum of discounted rewards recursively using the so-called state-value function [14]:

$$V(\boldsymbol{s}) = \min_{\boldsymbol{a}\in\boldsymbol{\mathcal{A}}} \left\{ J(\boldsymbol{s},\boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}'\in\boldsymbol{\mathcal{S}}} p(\boldsymbol{s}' \mid \boldsymbol{s},\boldsymbol{a}) V(\boldsymbol{s}') \right\}, \tag{10}$$

which can be computed offline, prior to run-time, using a well-known technique called value-iteration[8] (VAL_I) [14]: i.e.,

$$V^k(\boldsymbol{s}) = \min_{\boldsymbol{a}\in\boldsymbol{\mathcal{A}}} \left\{ J(\boldsymbol{s},\boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}'\in\boldsymbol{\mathcal{S}}} p(\boldsymbol{s}' \mid \boldsymbol{s},\boldsymbol{a}) V^{k-1}(\boldsymbol{s}') \right\} \tag{11}$$

where k is the iteration index and V^0 is an arbitrary initial value function. Given the optimal state-value function V^* (obtained as $k \to \infty$), the optimal policy π^* can be computed as:

$$\pi^*(\boldsymbol{s}) = \arg\min_{\boldsymbol{a}\in\boldsymbol{\mathcal{A}}} \left\{ J(\boldsymbol{s},\boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}'\in\boldsymbol{\mathcal{S}}} p(\boldsymbol{s}' \mid \boldsymbol{s},\boldsymbol{a}) V^*(\boldsymbol{s}') \right\} \tag{12}$$

To compute the optimal stationary policy using VAL_I, the transition probability function and cost functions must be known and stationary. Throughout this paper, for the centralized VAL_I and the layered VAL_I algorithms proposed in Section V, we assume that these conditions are satisfied; however, in Section VI, we discuss potential extensions to this work, which relax these assumptions, and allow the optimal policy to be determined online, at run-time.

Table I summarizes the information requirements, memory complexity, and offline computational complexity associated with the centralized solution. Notice that for a large number of states, the computational complexity of the centralized VAL_I algorithm can be huge. We will see that decoupling the decision processes at the various layers can dramatically reduce the complexity.

---

[7] If $\gamma = 0$, then (9) reduces to the immediate expected reward. If $\gamma \neq 0$, then (9) takes into account the system's long-run rewards, but geometrically discounts the future rewards by γ. Discounting is desirable because the multimedia session's lifetime is not known a priori (i.e. the session may end unexpectedly) and therefore rewards should be maximized sooner rather than later. In this context, the discount factor can be interpreted as the probability that the session will continue for another stage.

[8] Policy iteration can also be used [14].

Table I. Centralized MDP: Information requirements, memory complexity, and computational complexity.

	Transition probability function	Cost function
Information Requirements	$p(\boldsymbol{s}' \mid \boldsymbol{s},\boldsymbol{a})$	$J(\boldsymbol{s},\boldsymbol{a})$
Memory Complexity	$\lvert\boldsymbol{\mathcal{S}}\rvert^2 \lvert\boldsymbol{\mathcal{A}}\rvert$	$\lvert\boldsymbol{\mathcal{S}}\rvert\lvert\boldsymbol{\mathcal{A}}\rvert$
Offline Computational Complexity	$O(\lvert\boldsymbol{\mathcal{S}}\rvert^2 \lvert\boldsymbol{\mathcal{A}}\rvert)$ per VAL_I	
Performance Gap	Provides benchmark for optimal performance	

B. Decentralized Performance Gap

The centralized VAL_I algorithm presented in the previous subsection yields the centralized optimal policy π^*, which maps *joint* states to *joint* actions. However, as we described in the introduction, it may be impractical to make joint coding decisions for all of the layers. Hence, the goal of the decompositions proposed in the next section (i.e. Section V) is to determine local policies π_l^*, for all $l \in \boldsymbol{\mathcal{L}}$, which map the *available information* at layer l to *local* actions at layer l. We denote the resulting decentralized policy as $(\pi_1^*,\ldots,\pi_L^*)$.

The centralized optimal policy π^* provides a performance benchmark for the decentralized policy $(\pi_1^*,\ldots,\pi_L^*)$. We can analytically determine the expected performance of π^*, $E_{\pi^*}[J]$, as follows:

$$E_{\pi^*}[J] = \sum_{\boldsymbol{s}\in\boldsymbol{\mathcal{S}}} \mu_{\pi^*}(\boldsymbol{s}) J(\boldsymbol{s},\pi^*(\boldsymbol{s})), \tag{13}$$

where $\mu_{\pi^*}(\boldsymbol{s})$ is the stationary probability of being in state $\boldsymbol{s} \in \boldsymbol{\mathcal{S}}$ given the centralized policy π^*. Similarly, the expected performance of $(\pi_1^*,\ldots,\pi_L^*)$, $E_{(\pi_1^*,\ldots,\pi_L^*)}[J]$, can be determined as:

$$E_{(\pi_1^*,\ldots,\pi_L^*)}[J] = \sum_{\boldsymbol{s}\in\boldsymbol{\mathcal{S}}} \mu_{(\pi_1^*,\ldots,\pi_L^*)}(\boldsymbol{s}) J(\boldsymbol{s},(\pi_1^*(\boldsymbol{s}),\ldots,\pi_L^*(\boldsymbol{s}))). \tag{14}$$

where $\mu_{(\pi_1^*,\ldots,\pi_L^*)}(\boldsymbol{s})$ is the stationary probability of being in state $\boldsymbol{s} \in \boldsymbol{\mathcal{S}}$ given the decentralized policy $(\pi_1^*,\ldots,\pi_L^*)$. The *performance gap* between a decentralized policy $(\pi_1^*,\ldots,\pi_L^*)$ and the optimal centralized policy π^* is merely the difference between (14) and (13), i.e.

$$\text{Performance gap} = E_{(\pi_1^*,\ldots,\pi_L^*)}[J] - E_{\pi^*}[J]. \tag{15}$$

Since the decentralized solutions proposed in this paper are approximations of the centralized solution, the performance gap is always non-negative (i.e. $E_{\pi^*}[J] \leq E_{(\pi_1^*,\ldots,\pi_L^*)}[J]$).

Building on the decompositions proposed in this paper, future research should investigate new decompositions and models that can be deployed to minimize the performance gap.

C. Decomposition methodology

The illustrative decompositions presented in Section V all follow a general decomposition methodology. In Table II, we outline this methodology to connect all of the examples and to provide a template for future

research that deals with different data and budget dependency structures or different problem settings (e.g. Reconfigurable Video Coding).

Step 3 of the proposed decomposition methodology is about *decomposition principles*, which are used to formulate a decomposable approximation of a centralized VAL_I algorithm. In our illustrative examples, we define several useful decomposition principles; however, extensions to this work may identify new principles for use in different situations. Decomposition principles that lead to minimal performance gaps are desirable.

Step 4 yields the local value iteration algorithms at each layer. These local VAL_I algorithms must conform to the standard VAL_I form in (11) to guarantee that they converge to an optimal state-value function and a corresponding optimal policy for the given cost and transition probability functions [20].

Table II. Proposed decomposition methodology.

Step 1: Define problem-specific cost and transition probability functions	Additively decompose the cost function across layers and factor the transition probability function based on the problem-specific data and budget dependencies.
Step 2: Reformulate the centralized VAL_I algorithm	Substitute the additively decomposed cost function and factored transition probability function from step 1 into the centralized VAL_I algorithm.
Step 3: Apply appropriate decomposition principles	Identify the dependencies in the reformulated centralized VAL_I algorithm from step 2 and apply the appropriate decomposition principles to yield a decomposable approximation of the algorithm.
Step 4: Formulate local value iterations	Formulate local value iterations and determine the appropriate message exchanges to implement the decomposition and determine optimal local policies.

V. Decomposition Case Studies

In this subsection, we present three illustrative examples for decomposing the decision processes at each layer using a novel layered Markov decision process (MDP). We characterize the example decompositions by: (i) the considered dependencies; (ii) the information required to implement them; (iii) their memory and computational complexity; and, (iv) whether they perform as well as an optimal centralized solution, or if there is a performance gap introduced by decomposing the problem (e.g. due to imperfect models of one layer's impact on another's rate-distortion behavior).

A. Case 1: No dependencies

We first consider how to decompose the decision processes when there are neither data dependencies nor budget dependencies. This situation arises, for example, when using the simple multi-view coder described in Section II.C coupled with the IB buffer model described in Section III.A (see Fig. 5(a)).

Before we can decompose the decision process, however, we need to define the cost function J. In this setting, $J(\boldsymbol{s},\boldsymbol{a})$ can be additively decomposed into local costs associated with each view, i.e.

$$J(\boldsymbol{s},\boldsymbol{a}) = \sum_{l\in\boldsymbol{\mathcal{L}}}[\,g_l(s_l,a_l) + \lambda_l \cdot d_l(a_l)\,], \tag{16}$$

where the buffer cost at layer l, $g_l(s_l,a_l)$, is defined as in (33), with $f_l(\boldsymbol{a}(\Gamma_l)) = \mathbf{0}$ because there are no data dependencies, and λ_l weights the relative importance of each sequence's buffer cost with its distortion cost.

Using the factored transition probability function defined in (5) for the IB model (with $f_l(\boldsymbol{a}(\Gamma_l)) = \mathbf{0}$) and the additive cost function defined in (16), the centralized VAL_I algorithm defined in (11) can be rewritten as

$$V^k(\boldsymbol{s}) = \min_{\boldsymbol{a}\in\boldsymbol{\mathcal{A}}}\left\{\begin{array}{l}\displaystyle\sum_{l\in\boldsymbol{\mathcal{L}}}[\,g_l(s_l,a_l) + \lambda_l \cdot d_l(a_l)\,] \\ \displaystyle +\gamma\sum_{\boldsymbol{s}'\in\boldsymbol{\mathcal{S}}}\prod_{l=1}^{L} p(s_l' \mid s_l,a_l)V^{k-1}(\boldsymbol{s}')\end{array}\right\}, \tag{17}$$

which can be decomposed into independent VAL_Is at each layer $l \in \boldsymbol{\mathcal{L}}$, i.e.

$$V_l^k(s_l) = \min_{a_l\in\mathcal{A}_l}\left\{\begin{array}{l} g_l(s_l,a_l) + \lambda_l \cdot d_l(a_l) \\ \displaystyle +\gamma\sum_{s_l'\in S_l} p(s_l' \mid s_l,a_l)V_l^{k-1}(s_l')\end{array}\right\}. \tag{18}$$

Given its optimal local state-value function V_l^* (obtained as $k\to\infty$), layer l can compute its locally optimal policy π_l^* offline as:

$$\pi_l^*(s_l) = \arg\min_{a_l\in\mathcal{A}_l}\left\{\begin{array}{l} g_l(s_l,a_l) + \lambda_l \cdot d_l(a_l) \\ \displaystyle +\gamma\sum_{s_l'\in S_l} p(s_l' \mid s_l,a_l)V_l^*(s_l')\end{array}\right\}. \tag{19}$$

Notice that $V^*(\boldsymbol{s}) = \sum_{l\in\boldsymbol{\mathcal{L}}} V_l^*(s_l)$ and $\pi^*(\boldsymbol{s}) = (\pi_1^*(s_1),\ldots,\pi_L^*(s_L))$. This leads us to our first decomposition principle:

Decomposition principle 1: If there are no dependencies among the layers (i.e. neither data nor budget dependencies), then the problem can be decomposed without performance loss (i.e. the performance gap defined in (15) is zero).

Table I summarizes the information requirements, memory complexity, and offline computational complexity associated with the decomposition with no dependencies.

Table III. Decomposition information: No dependencies.

	Transition probability function	Cost function
Information Requirements	$p(s_l' \mid s_l,a_l),\ \forall l\in\boldsymbol{\mathcal{L}}$	$g_l(s_l,a_l),\ d_l(a_l),\ \forall l\in\boldsymbol{\mathcal{L}}$
Memory Complexity	$\sum_{l\in\boldsymbol{\mathcal{L}}} \lvert\mathcal{S}_l\rvert^2\lvert\mathcal{A}_l\rvert$	$\lvert\mathcal{S}_l\rvert\lvert\mathcal{A}_l\rvert + \lvert\mathcal{A}_l\rvert$
Offline Computational Complexity	$O(\lvert\mathcal{S}_l\rvert^2\lvert\mathcal{A}_l\rvert)$ per VAL_I for all $l\in\boldsymbol{\mathcal{L}}$	
Performance Gap	No performance gap when there are no dependencies among layers	

B. Case 2: Data dependences only

In our second case study, we consider how to decompose the decision processes when there are data dependencies but no budget dependencies. This situation arises, for example, when using the scalable coder described in Section II.A (see Fig. 3) coupled with the IB buffer model described in Section III.A (see Fig. 5(a)).

In this setting, the cost function J is similar to when there are no dependencies, however, the reference frames $f_l(\boldsymbol{a}(\Gamma_l)) = f_l(a_1,\ldots,a_{l-1})$ cannot be ignored due to the data dependencies among the various layers. Hence, $J(\boldsymbol{s},\boldsymbol{a})$ can be additively decomposed as follows:

$$\begin{aligned} J(\boldsymbol{s},\boldsymbol{a}) = {} & g_1(s_1,a_1) + \lambda_1 \cdot d_1(a_1) \\ & + \sum_{l=2}^{L} [\, g_l(s_l,a_l,f_l(a_1,\ldots,a_{l-1})) - \lambda_l \cdot \Delta d_l(a_l,f_l(a_1,\ldots,a_{l-1}))], \end{aligned} \tag{20}$$

where $g_l(s_l,a_l,f_l(a_1,\ldots,a_{l-1}))$, for all $l \in \boldsymbol{\mathcal{L}}$, is defined as in (33); λ_1 weights the relative importance of the base layer's buffer cost and its distortion; and, λ_l, for $l \in \{2,\ldots,L\}$, weights the relative importance of each enhancement layer's buffer cost with its distortion reduction.

Using the factored transition probability function defined in (5) for the IB model and the additive cost function defined in (20), the centralized VAL_I algorithm defined in (11) can be rewritten as

$$V^k(\boldsymbol{s}) = \min_{\boldsymbol{a}\in\boldsymbol{\mathcal{A}}} \left\{ \begin{array}{l} g_1(s_1,a_1) + \lambda_1 \cdot d_1(a_1) \\ + \sum_{l=2}^{L} [\, g_l(s_l,a_l,f_l(a_1,\ldots,a_{l-1})) - \lambda_l \cdot \Delta d_l(a_l,f_l(a_1,\ldots,a_{l-1}))] \\ + \gamma \sum_{\boldsymbol{s}'\in\boldsymbol{\mathcal{S}}} \prod_{l=1}^{L} p\big(s_l' \mid s_l,a_l,f_l(a_1,\ldots,a_{l-1})\big) V^{k-1}\big(\boldsymbol{s}'\big) \end{array} \right\}. \tag{21}$$

Unfortunately, because the enhancement layers depend on the reference frames $f_l(a_1,\ldots,a_{l-1})$, for $l \in \{2,\ldots,L\}$, (21) cannot be decomposed optimally like (17) (i.e. with a zero performance gap).

We invoke the following decomposition principle to decompose (21) into local VIs at each layer:

Decomposition principle 2: If a descendent layer's optimal actions depend on the reference frames generated by its ancestor layers, or their actions, then the decision processes of the layers can be decomposed by allowing the descendent layers to *abstract/model* the reference frames or actions of the ancestor layers.

A good abstraction is some quantity, or set of quantities, that is well correlated with the actions taken by the ancestor layers and with the rate-distortion behavior of the descendent layers. Better models, which incorporate more information, will decrease the performance gap, at the expense of increased memory and

computational complexity.

In this instance, we propose to use the distortion $d_{\Lambda_{l-1}}(a_1,\ldots,a_{l-1})$ as an abstraction of the reference frames $f_l(a_1,\ldots,a_{l-1})$. Importantly, other reference descriptions could be used instead of, or in addition to, the distortion (e.g. the number of motion vectors or their magnitudes); however, as stated in the second decomposition principle, richer models increase the decomposition's complexity.

Because $d_{\Lambda_{l-1}}(a_1,\ldots,a_{l-1})$ is continuous, we quantize it to determine the *reference distortion description* (RDD) $x_l = \psi_d\left(d_{\Lambda_{l-1}}(a_1,\ldots,a_{l-1})\right)$, where $\psi_d : \mathbb{R}_+ \mapsto \mathcal{X}$ is a quantizer with reconstruction values in the set $\mathcal{X}$. Substituting x_l for $f_l(a_1,\ldots,a_{l-1})$, (21) can be approximated as

$$V^k(\boldsymbol{s}) \cong \min_{\boldsymbol{a}\in\boldsymbol{\mathcal{A}}} \left\{ \begin{array}{l} g_1(s_1,a_1) + \lambda_1 \cdot d_1(a_1) + \sum_{l=2}^{L} [\, g_l(s_l,a_l,x_l) - \lambda_l \cdot \Delta d_l(a_l,x_l)\,] \\ +\gamma \sum_{\boldsymbol{s}'\in\boldsymbol{\mathcal{S}}} \prod_{l=1}^{L} p\left(s_l' \mid s_l,a_l,x_l\right) V^{k-1}\left(\boldsymbol{s}'\right) \end{array} \right\}, \qquad (22)$$

We propose two solutions for decomposing (22). The first solution allows each layer to independently determine its optimal policy, using only limited information from its immediate ancestor. We refer to this solution as the *open-loop decomposition* because there is no information sent from the descendent layers to the ancestor layers to indicate to them how their decisions impact the costs at the descendent layers. In contrast, the second solution allows information to be propagated from the ancestor layers to the descendent layers and vice versa. We refer to this solution as the *closed-loop decomposition*.

1) Open-loop decomposition

The open-loop decomposition begins with the base layer determining its locally optimal stationary policy $\pi_1^* : \mathcal{S}_1 \mapsto \mathcal{A}_1$ independently of the states and actions at layers $l \in \{2,\ldots,L\}$. It does this using the following offline local VAL_I algorithm:

$$V_1^k(s_1) = \min_{a_1\in\mathcal{A}_1} \left\{ \begin{array}{l} g_1(s_1,a_1) + \lambda_1 \cdot d_1(a_1) \\ +\gamma \sum_{s_1'} p\left(s_1' \mid s_1,a_1\right) V_1^{k-1}\left(s_1'\right) \end{array} \right\}. \qquad (23)$$

After computing its optimal local state-value function V_1^* (obtained as $k \to \infty$ in (23)), the base layer can compute its locally optimal stationary policy π_1^* as:

$$\pi_1^*(s_1) = \arg\min_{a_1\in\mathcal{A}_1} \left\{ \begin{array}{l} g_1(s_1,a_1) + \lambda_1 \cdot d_1(a_1) \\ +\gamma \sum_{s_1'\in S_1} p\left(s_1' \mid s_1,a_1\right) V_1^*\left(s_1'\right) \end{array} \right\}. \qquad (24)$$

Recall from (22) that the transition probability and cost functions at layer $l \in \{2,\ldots,L\}$ depend on the RDD x_l from layer $l-1$. Hence, for layers $l \in \{2,\ldots,L\}$, we define a composite state $\vec{s}_l = (s_l, x_l)$. Based

on its composite state, layer $l \in \{2,\ldots,L\}$ performs its local VAL_I offline as follows:

$$V_l^k(s_l,x_l) = \min_{a_l \in \mathcal{A}_l} \left\{ \begin{array}{l} g_l(s_l,a_l,x_l) - \lambda_l \cdot \Delta d_l(a_l,x_l) \\ \\ +\gamma \sum_{s_l' \in \mathcal{S}_l} p\left(s_l' \mid s_l,a_l,x_l\right) \sum_{x_l' \in \mathcal{X}} p\left(x_l'\right) V_l^{k-1}\left(s_l',x_l'\right) \end{array} \right\}, \tag{25}$$

where $p\left(x_l'\right)$ is the stationary probability of the RDD $x_l' \in \mathcal{X}$ from layer $l-1$. The distribution $\{p(x_l) \mid x_l \in \mathcal{X}\}$, for $l \in \{2,\ldots,L\}$, can be calculated by layer $l-1$ after it determines its optimal stationary policy π_{l-1}^*. Specifically,

$$p(x_l) = \sum_{s_{l-1} \in \mathcal{S}_{l-1}} \mu_{\pi_{l-1}^*}(s_{l-1}) I\left[\psi_d\left(d_{\Lambda_{l-1}}\left(\pi_{l-1}^*(s_{l-1})\right)\right) = x_l\right], \tag{26}$$

where $\mu_{\pi_{l-1}^*}(s_{l-1})$ is the stationary probability that layer $l-1$ is in state s_{l-1} given policy π_{l-1}^*, and $I\left[\psi_d\left(d_{\Lambda_{l-1}}\left(\pi_{l-1}^*(s_{l-1})\right)\right) = x_l\right]$ is an indicator function that takes value 1 when $\psi_d\left(d_{\Lambda_{l-1}}\left(\pi_{l-1}^*(s_{l-1})\right)\right) = x_l$, and takes value 0 otherwise. Hence, after layer $l-1$ performs its local VAL_I, it must calculate the distribution $\{p(x_l) \mid x_l \in \mathcal{X}\}$ using (26) and forward it to layer l for use in its local VAL_I. As we mentioned before, other reference descriptions could be used in addition to the distortion (e.g. the number of motion vectors or their magnitudes) leading to more sophisticated models than (26).

Finally, after computing its optimal local state-value function V_l^* offline (obtained as $k \to \infty$ in (25)), layer $l \in \{2,\ldots,L\}$ can compute its locally optimal policy as

$$\pi_l^*(s_l,x_l) = \arg\min_{a_l \in \mathcal{A}_l} \left\{ \begin{array}{l} g_l(s_l,a_l,x_l) - \lambda_l \cdot \Delta d_l(a_l,x_l) \\ \\ +\gamma \sum_{s_l'} p\left(s_l' \mid s_l,a_l,x_l\right) \sum_{x_l'} p\left(x_l'\right) V_l^*\left(s_l',x_l'\right) \end{array} \right\}, \tag{27}$$

where $\pi_l^*(s_l,x_l)$ dictates the optimal action a_l^* to take in state s_l given the RDD x_l from layer $l-1$.

Table IV summarizes the information requirements, memory complexity, and offline computational complexity associated with the open-loop decomposition with only data dependencies, and describes why it incurs a performance gap relative to the optimal centralized policy. We observe that the computational complexity in this setting is significantly smaller than for the centralized case as long as the number of RDDs is not too large. Fig. 6 illustrates how each layer computes its optimal local policy in the open-loop decomposition, and what information must be shared among layers.

Table IV. Decomposition information: Open-loop decomposition with only data dependencies.

	Transition probability function	Cost function
Information Requirements	$p(s_1' \mid s_1, a_1)$, $l = 1$ $p(s_l' \mid s_l, a_l, x_l)$, $p(x_l)$, o.w.	$g_1(s_1, a_1)$, $d_1(a_1)$, $l = 1$ $g_l(s_l, a_l, x_l)$, $\Delta d_l(a_l, x_l)$, o.w.
Memory Complexity	$\lvert\mathcal{S}_1\rvert^2 \lvert\mathcal{A}_1\rvert + \sum_{l=2}^{L} \left(\lvert\mathcal{S}_l\rvert^2 \lvert\mathcal{A}_l\rvert \lvert\mathcal{X}_l\rvert + \lvert\mathcal{X}_l\rvert \right)$	$\lvert\mathcal{S}_1\rvert \lvert\mathcal{A}_1\rvert + \lvert\mathcal{A}_1\rvert + \sum_{l=2}^{L} (\lvert\mathcal{S}_l\rvert \lvert\mathcal{A}_l\rvert \lvert\mathcal{X}_l\rvert + \lvert\mathcal{A}_l\rvert \lvert\mathcal{X}_l\rvert)$
Offline Computational Complexity	$O\left(\lvert\mathcal{S}_1\rvert^2 \lvert\mathcal{A}_1\rvert\right)$, per VAL_I at layer $l = 1$ $O\left(\lvert\mathcal{S}_l \times \mathcal{X}_l\rvert^2 \lvert\mathcal{A}_l\rvert\right)$, per VAL_I o.w.	
Performance Gap	(i) Performance gap incurred because the descendent layers imperfectly model the ancestor layers based on the RDD. (ii) Performance gap incurred because the ancestor layers do not know their impact on the descendent layers.	

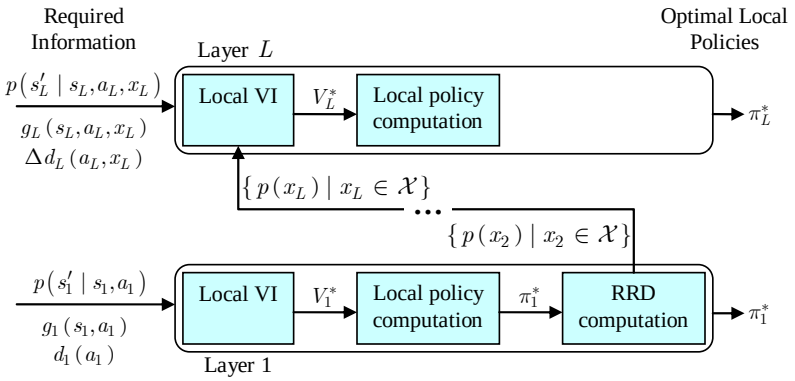


Fig. 6. Open-loop decomposition for scalable video coding with only data dependencies.

2) Closed-loop decomposition

In the open-loop decomposition, the ancestor layers do not consider how their decisions impact the costs and state transitions at the descendant layers. Consequently, the optimal local policies $(\pi_1^*, \ldots, \pi_L^*)$ derived using the open-loop decomposition can be suboptimal compared to the centralized optimal policy π^* under the conditions described in Appendix A. Our proposed solution to this problem is our third decomposition principle:

Decomposition principle 3: If the ancestor layers maintain a model of the impact of their actions on the descendant layers' costs and state transitions, then this model can be integrated into the ancestor layer's VAL_I algorithm to improve the decentralized decision policy.

For illustration, let us assume that there are only two layers and that the base layer (i.e. $l = 1$) knows the joint state $\boldsymbol{s} = (s_1, s_2)$. Let the base layer have a model $M(s_2, a_1) = \left(J_M(s_2, a_1), p_M\left(s_2' \mid s_2, a_1\right)\right)$ of how its actions impact the enhancement layer's cost and state transition given the enhancement layer's state. Given the model M, the VAL_I at the base layer can be performed offline as follows:

$$V_1^k(\boldsymbol{s}) = \min_{a_1 \in \mathcal{A}_1} \left\{ \begin{array}{l} g_1(s_1,a_1) + \lambda_1 \cdot d_1(a_1) + J_M(s_2,a_1) \\ +\gamma \sum_{\boldsymbol{s}' \in \boldsymbol{\mathcal{S}}} p(s_1' \mid s_1,a_1)\, p_M(s_2' \mid s_2,a_1) V_1^{k-1}(\boldsymbol{s}') \end{array} \right\}. \tag{28}$$

After computing V_1^* (obtained as $k \to \infty$), the base layer can determine its optimal local policy π_1^* as follows:

$$\pi_1^*(\boldsymbol{s}) = \arg\min_{a_1 \in \mathcal{A}_1} \left\{ \begin{array}{l} g_1(s_1,a_1) + \lambda_1 \cdot d_1(a_1) + J_M(s_2,a_1) \\ +\gamma \sum_{\boldsymbol{s}' \in \boldsymbol{\mathcal{S}}} p(s_1' \mid s_1,a_1)\, p_M(s_2' \mid s_2,a_1) V_1^{k}(\boldsymbol{s}') \end{array} \right\} \tag{29}$$

Subsequently, the closed-loop decomposition proceeds like the open-loop decomposition illustrated in Fig. 6: i.e., the base layer computes its RDD distribution $\{p(x_2) \mid x_2 \in \mathcal{X}\}$, and forwards it to the enhancement layer, which subsequently performs its own local VAL_I offline using (25).

It remains to explain how the base layer can determine $J_M(s_2,a_1)$ and $p_M(s_2' \mid s_2,a_1)$. These models can be derived by allowing the enhancement layer to share its cost function $g_2(s_2,a_2,x_2) - \lambda_2 \cdot \Delta d_2(a_2,x_2)$ and transition probability function $p(s_2' \mid s_2,a_2,x_2)$ with the base layer. Then, given these functions, the base layer can estimate its models by substituting the distortion description x_2 with the actual expected distortion description $\psi_d(d_1(a_1))$ given action a_1: i.e.,

$$\begin{aligned} J_M(s_2,a_1) &= g_2(s_2,a_2,\psi_d(d_1(a_1))) - \lambda_2 \cdot \Delta d_2(a_2,\psi_d(d_1(a_1))) \\ p_M(s_2' \mid s_2,a_1) &= p(s_2' \mid s_2,a_2,\psi_d(d_1(a_1))). \end{aligned} \tag{30}$$

Unfortunately, plugging in these models into the base layer's local VAL_I defined in (28) *does not decouple the decision processes* at the two layers because (30) still depends on the action at the enhancement layer (i.e. a_2). Hence, without further modification to (30), we cannot determine the optimal stationary policy π_1^* at the base layer without knowing the optimal stationary policy π_2^* at the enhancement layer (which would fix action a_2 for each s_2, i.e. $a_2^* = \pi_2^*(s_2)$, thereby making (30) independent of a_2), which in turn can only be determined if we know π_1^* as illustrated in the previously described open-loop decomposition.

To decouple the interdependent decision processes, we must modify (30) to be independent of the enhancement layer's action. We assume that each action $a_2 \in \mathcal{A}_2$ is taken with probability $p_{\mathcal{A}_2}(a_2)$ (which may be derived from a previous encoding session), and therefore the models in (30) can be rewritten as

$$\begin{aligned} J_M(s_2,a_1) &= \sum_{a_2 \in \mathcal{A}_2} p_{\mathcal{A}_2}(a_2)[g_2(s_2,a_2,\psi_d(d_1(a_1))) - \lambda_2 \cdot \Delta d_2(a_2,\psi_d(d_1(a_1)))] \\ p_M(s_2' \mid s_2,a_1) &= \sum_{a_2 \in \mathcal{A}_2} p_{\mathcal{A}_2}(a_2)\, p(s_2' \mid s_2,a_2,\psi_d(d_1(a_1))), \end{aligned} \tag{31}$$

which are independent of a_2. Although this is not a perfect model, it is indicative of the *relative* impact of the base layer's various actions on the enhancement layer's cost and state transition. Note that it is also possible to

simplify (31) by averaging out s_2 in addition to a_2.

Table V summarizes the information requirements, memory complexity, and offline computational complexity associated with the closed-loop decomposition with only data dependencies, and describes why it incurs a performance gap relative to the optimal centralized policy. Fig. 7 illustrates how each layer computes its optimal local policy in the closed-loop decomposition, and what information must be shared among layers.

Table V. Decomposition information: Closed-loop decomposition with only data dependencies.

	Transition probability function	Cost function
Information Requirements	$p(s_1' \mid s_1, a_1)$, $p_M(s_2' \mid s_2, a_1)$, $l=1$ $p(s_2' \mid s_2, a_2, x_2)$, $p(x_2)$, $l=2$	$g_1(s_1, a_1)$, $d_1(a_1)$, $J_M(s_2, a_1)$, $l=1$ $g_2(s_2, a_2, x_2)$, $\Delta d_2(a_2, x_2)$, $l=2$
Memory Complexity	$\lvert\mathcal{S}_1\rvert^2 \lvert\mathcal{A}_1\rvert + \lvert\mathcal{S}_2\rvert^2 \lvert\mathcal{A}_1\rvert +$ $\lvert\mathcal{S}_2\rvert^2 \lvert\mathcal{A}_2\rvert\lvert\mathcal{X}_2\rvert + \lvert\mathcal{X}_2\rvert$	$\lvert\mathcal{S}_1\rvert\lvert\mathcal{A}_1\rvert + \lvert\mathcal{A}_1\rvert + \lvert\mathcal{S}_2\rvert\lvert\mathcal{A}_1\rvert +$ $\lvert\mathcal{S}_2\rvert\lvert\mathcal{A}_2\rvert\lvert\mathcal{X}_2\rvert + \lvert\mathcal{A}_2\rvert\lvert\mathcal{X}_2\rvert$
Offline Computational Complexity	$O\left(\lvert\boldsymbol{\mathcal{S}}\rvert^2 \lvert\mathcal{A}_1\rvert\right)$, per VAL_I at layer $l=1$ $O\left(\lvert\mathcal{S}_2 \times \mathcal{X}_2\rvert^2 \lvert\mathcal{A}_2\rvert\right)$, at layer $l=2$	
Performance Gap	(i) Performance gap incurred because the descendent layers imperfectly model the ancestor layers based on the RDD. (ii) Performance gap incurred because the ancestor layers imperfectly model their impact on the descendent layers using $J_M(s_2, a_1)$ and $p_M(s_2' \mid s_2, a_1)$.	

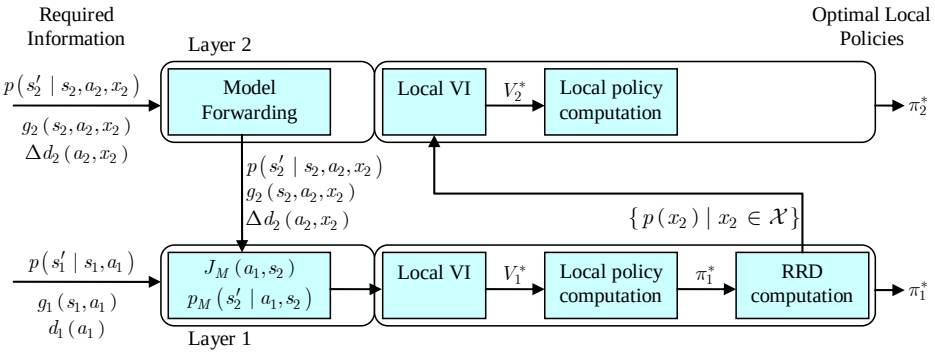


Fig. 7. Closed-loop decomposition for scalable video coding with only data dependencies.

C. Case 3: Data and Budget Dependencies

In our third case study, we consider how to decompose the decision processes when there are data dependencies and budget dependencies. This situation arises, for example, when using the scalable coder described in Section II.A (see Fig. 3) coupled with the SBIC buffer model described in Section III.B (see Fig. 5(b)) or the SBJC buffer model described in Section III.C (see Fig. 5(c)).

Due to its similarities to the previous case study with only data dependencies, we discuss this case study in Appendix C.

VI. Extensions of the Proposed Framework

A. Applications to Reconfigurable Video Coding

Reconfigurable Video Coding (RVC) MPEG decoders can dynamically compose decoding solutions from a set of registered video coding tools and libraries [9] [10]. To fully take advantage of this flexibility at the decoder, an encoder mechanism must be in place for optimally selecting among the available coding tools depending on the video source characteristics and the coupled behavior of the interacting components. For example, a particular transform-quantizer pair may work best for natural video content, while another may be best for computer generated video content.

Fig. 8 illustrates an example RVC-based encoder, where each codec component has several possible instantiations. The codec components correspond to functional blocks in the encoder pipeline (e.g. transform, quantization, motion estimation, entropy coding) and each component instantiation corresponds to a different implementation of that functional block from a different standard or company. A distributed control plane computes the optimal local policies offline; then, online, at run-time, the codec components select their instantiations based on their offline computed policies and the state feedback.

Although there are several differences between the RVC-based encoder and the layered encoder, there are also several marked similarities. First, in the layered encoder, actions at the ancestor layers impact the rate-distortion performance of the descendent layers. In the RVC-based encoder, on the other hand, actions at the earlier components impact the performance of the later components (e.g. the choice of transform impacts the quantizer's efficiency). Second, in the layered encoder, actions act on DUs in each layer. In the RVC-based encoder, on the other hand, actions act on the same DU, but across different components. Finally, in the layered encoder, post-encoding buffers contain bits. In the RVC-based encoder, on the other hand, the buffers between components contain processed data. Despite these differences, the decisions among layers or components are coupled, which allow us to apply a similar decomposition methodology as the one in this paper to the coding tool selection problem in RVC-based encoders.

The proposed framework provides a methodology for decoupling the decision processes of two or more codec component functions, by enabling them to autonomously select their optimal instantiations in response to the behavior of their interconnected components. Using the terminology introduced in this paper, both open-loop and closed-loop decompositions are possible as illustrated in Fig. 8. An open-loop decomposition would begin with the first component computing its optimal local policy and then forwarding information about its

selected operating points to the second component, which will then compute its optimal local policy and forward information to the third component, and so forth, with each component ignoring its impact on the subsequent components when determining its optimal local policy. In contrast, the closed-loop decomposition would begin with the earlier components modeling their impact on the latter components, and then proceeding as in the open-loop decomposition, but with modified local VAL_I algorithms that account for the impact of the first components on the latter components.

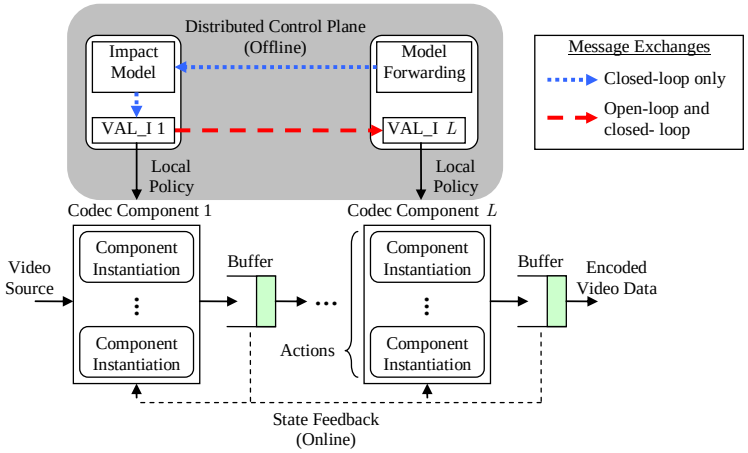


Fig. 8. Example RVC-based encoder and corresponding distributed control plane.

B. Dynamic online adaptation

In this paper, we assume that the rate-distortion behavior of each layer is known and stationary. This enables us to determine the cost and state transition probability functions used in the various decompositions (i.e. the "required information" in Table III, Table IV, Table V, Table VIII, and Table IX). In practice, however, the rate-distortion behavior is unknown a priori and therefore the transition probability and cost functions are also unknown. Consequently, online learning techniques [13][14] must be deployed to learn the optimal policies for each layer at run-time, based on experience.

One possible learning solution is to estimate the cost and transition probability functions using maximum likelihood estimation as new experience is acquired, and then recompute the optimal local policies at each layer based on the estimated quantities. Unfortunately, this technique can incur significant computational overheads every time the policy is updated (see the "computational complexity" fields in Table III, Table IV, Table V, Table VIII, and Table IX for order-of-magnitude estimates).

An alternative, and preferable, solution is to deploy low-complexity reinforcement learning techniques that are explicitly designed for learning MDP policies online [13] [14] [17]. Reinforcement learning techniques can

be used to directly update the state-value function and policy after encoding each DU.

Fig. 9 illustrates a conceptual reinforcement learning implementation when there are two layers with coupled state transitions and costs. The learning process works as follows:

- Given their current state information, each layer takes an action dictated by its current policy.
- These actions result in costs incurred at each layer, and a state transition, both of which may depend on the states and actions at the other layer because of the coupled dynamics.
- The incurred costs are then used to update the value function and policy at each layer, and the process repeats.

Similar to the decompositions proposed in this paper, learning decompositions may also incur performance loss compared to a centralized learning algorithm. Investigating these learning techniques forms an important area of future research.

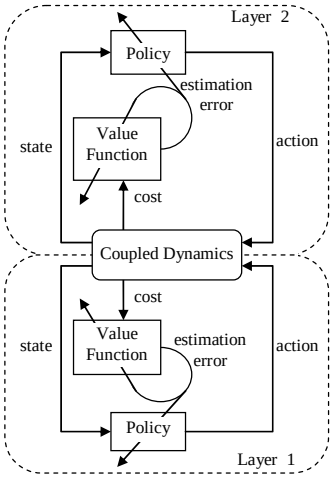


Fig. 9. Conceptual reinforcement learning diagram for two layers.

VII. Illustrative Results

In this section, we compare the performance of the proposed open-loop and closed-loop decompositions to the performance of the centralized solutions using the IB and SBIC buffer models. For illustration, we use the H.264 JM Reference Encoder (version 13.2) to generate scalable bitstreams of *Foreman* (first 145 frames, CIF resolution, low motion) and *Stefan* (last 145 frames, CIF resolution, high motion) with two temporal layers. The low frame-rate base layer contains only I and P frames (15 frames per second, intra-period 8) and the enhancement layer contains only B frames (which contribute an additional 15 frames per second when decoded).

A. Illustrative layered coding system parameters

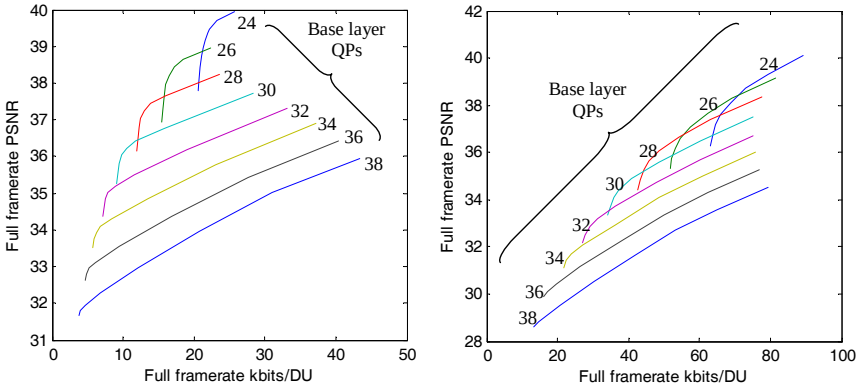
Table VI details the parameters used in our illustrative results for each buffer model, including information about the buffer sizes, number of buffer states, buffer drain rates, and action sets at each layer. In Table VI, parameters in parentheses are used for encoding *Stefan* and those that are not in parentheses are used for encoding *Foreman*. If no parentheses are present in a field, then the parameters are used for both *Stefan* and *Foreman*. We make the following observations about the selected parameters:

- For both buffer models, the buffer sizes and number of buffer states are selected to make each buffer state correspond to a 20 kbit interval for *Foreman* and a 40 kbit for *Stefan* after applying the state description mapping function defined in Section III.A. For illustration, we assume that the state description mapping function is a uniform quantizer such that each state represents an equal length interval of the buffer; however, it is possible that performance could be improved by using a non-uniform quantizer (e.g. using denser state descriptions for higher buffer occupancy levels).
- The enhancement layer's buffer under the SBIC model is as large as both buffers combined in the IB model. This is because the SBIC model imposes independent constraints on every independently decodable subset of layers (e.g. $\Lambda_1 = \{1\}$ and $\Lambda_2 = \{1,2\}$ in our example with two temporal layers), and therefore the enhancement buffer must simultaneously regulate the bits from the base layer and the enhancement layer.
- For both buffer models, the base layers' drain rates are the same and the total rates are the same (i.e. $b_{\Lambda_1} = \overline{b}_1$ and $b_{\Lambda_2} = \overline{b}_1 + \overline{b}_2$).

Fig. 10 illustrates the impact of the base layer's QP on the overall quality-rate performance of the video encoder for the Foreman and Stefan sequences. We observe that decreasing the base layer's QP increases the minimum bit-rate required to encode the full framerate sequence, and concomitantly decreases the bits required to encode the enhancement layer. In the context of Fig. 10, the goal of the proposed decentralized decision making framework is for the layers to autonomously configure themselves given the their buffer-constraints to select an optimal feasible operating point within the convex set defined by these curves.

Table VI. Parameters used for illustrative results.

		Base Layer $l = 1$	Enhancement Layer $l = 2$
IB Buffer Model	Buffer Size (kbits) $q_l^{\max}$	200 (400)	100 (200)
	No. Buffer States (uniform quantization)	10	5
	Drain Rate (kbits/DU) $\bar{b}_l$	40, 35, 25, 20, 15, 12.5, 10, 7.5 (125, 110, 90 70 60 45 30 25)	10 (60)
SBIC Buffer Model	Buffer Size (kbits) $q_{\Lambda_l}^{\max}$	$q_1^{\max}$	$q_1^{\max} + q_2^{\max}$
	No. Buffer States (uniform quantization)	10	15
	Drain Rate (kbits/DU) $\bar{b}_{\Lambda_l}$	$\bar{b}_1$	$\bar{b}_1 + \bar{b}_2$
	Action Sets (Encoder QPs) $\mathcal{A}_l$	$\{24, 26, 28, 30, 32, 34, 36, 38\}$	$\{24, 26, 28, 30, 32, 34, 36, 38\}$



(a) Foreman (b) Stefan

Fig. 10. Full framerate PSNR vs. Full framerate kbits/DU. (a) Foreman; (b) Stefan.

B. Value Iteration Complexity

Table VII compares the complexity of the centralized, open-loop, and closed-loop solutions for the IB and SBIC buffer models when using the parameters in Table VI[9]. We assume that the enhancement layer uses four reference distortion descriptions (see Section V.B) and four reference rate descriptions (see Appendix C) to model the base layer in the appropriate decompositions. The complexity per value iteration in all of the cases is normalized by $\mathcal{C}_{\text{IB}}$, which is the complexity of a centralized VAL_I using the IB buffer model.

[9] The data in Table VII is a specific instantiation of the "computational complexity" fields in Table I (centralized), Table IV (open-loop, IB buffer model), Table V (closed-loop, IB buffer model), Table VIII (open-loop, SBIC buffer model), and Table IX (closed-loop, SBIC buffer model) for the parameters in Table VI.

We observe from Table VII that decomposing the original centralized problem for each buffer model significantly reduces the VAL_I complexity by up to two orders of magnitude; however, both SBIC buffer model decompositions are still quite complex. This is because the enhancement layer must use both the RDDs and the RRDs to model the base layer, which results in exponential growth of its composite state set. Complexity can be reduced in all cases, at the expense of performance, by using a coarser quantization of the buffer states and/or using less RDDs and RRDs.

Table VII. Normalized complexity per value iteration using the illustrative parameters in Table VI. Four reference distortion descriptions and four reference rate descriptions are assumed.

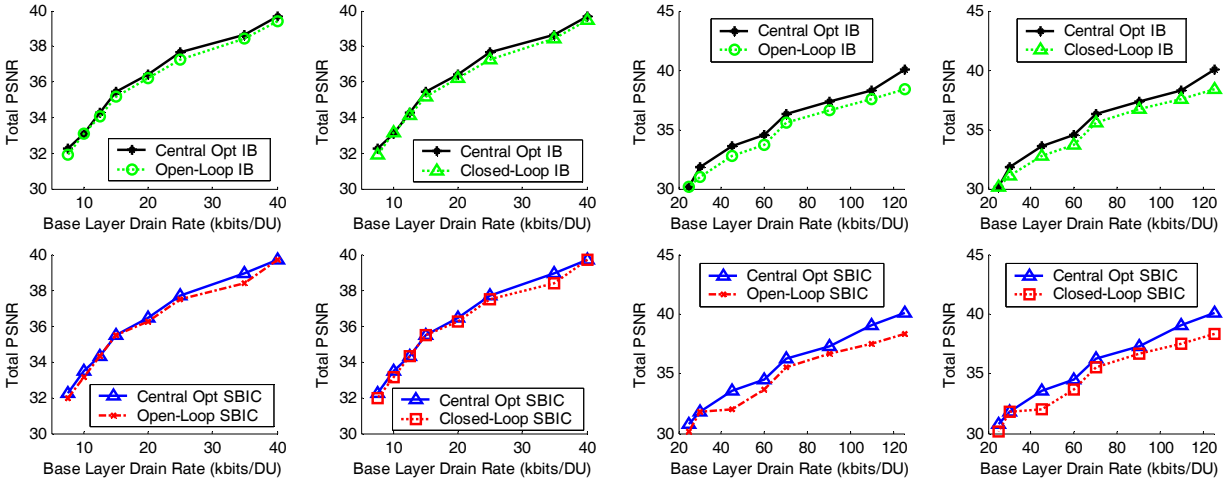
	Centralized		Open-Loop Decomposition	Closed-loop Decomposition
IB Buffer Model	$\mathcal{C}_{\mathrm{IB}}$ per VAL_I	Layer $l=1$	$0.005\mathcal{C}_{\mathrm{IB}}$ per VAL_I	$0.125\mathcal{C}_{\mathrm{IB}}$ per VAL_I
		Layer $l=2$	$0.02\mathcal{C}_{\mathrm{IB}}$ per VAL_I	$0.02\mathcal{C}_{\mathrm{IB}}$ per VAL_I
SBIC Buffer Model	$9\mathcal{C}_{\mathrm{IB}}$ per VAL_I	Layer $l=1$	$0.005\mathcal{C}_{\mathrm{IB}}$ per VAL_I	$1.125\mathcal{C}_{\mathrm{IB}}$ per VAL_I
		Layer $l=2$	$2.88\mathcal{C}_{\mathrm{IB}}$ per VAL_I	$2.88\mathcal{C}_{\mathrm{IB}}$ per VAL_I

C. Decomposition Performance Comparison

Fig. 11 compares the rate-distortion performance of the centralized, open-loop, and closed-loop solutions for the IB and SBIC buffer models when using the parameters in Table VI for the *Foreman* and *Stefan* sequences. Fig. 11(a) shows that decomposing the decision process for the relatively low motion *Foreman* sequence results in a performance gap (relative to the centralized optimal solution for the same buffer model) of less than 0.55 dB PSNR across all base layer drain rates using either buffer model. Meanwhile, Fig. 11(b) shows that decomposing the decision process for the high motion *Stefan* sequence results in a performance gap of less than 1.7 dB PSNR in all cases. These results indicate that the performance of the proposed solutions is highly dependent on the video source characteristics, which impact the ability of the layers to model each other.

We also observe from Fig. 11 that at high base layer drain rates, the optimal performance using the SBIC buffer model is better than the optimal performance using the IB buffer model (for the *Foreman* sequence, the maximum observed improvement is approximately 0.4 dB PSNR and for the *Stefan* sequence it is approximately 0.7 dB PSNR). This is because, in the former case, the enhancement layer can borrow bits from

the base layer to optimally balance the allocation of bits across layers.



(a) Foreman (b) Stefan

Fig. 11. Full framerate PSNR vs. Base layer's rate constraint (kbits/DU). (a) *Foreman* sequence (fixed enhancement drain rate of 10 kbits/DU); (b) *Stefan* sequence (fixed enhancement drain rate of 60 kbits/DU).

VIII. CONCLUSION

In this paper, we propose a framework for autonomous and foresighted decision making in layered video coders, which decouples the decision making processes at the various layers using a novel layered Markov decision process. We propose a step-by-step decomposition methodology for converting centralized MDP based problem formulations into decomposable layered MDPs. We apply this methodology in several illustrative settings with different data and budget dependencies among layers. In our illustrative results, we show that compared to a centralized MDP based solution, the proposed layered MDP decreases the optimization complexity by up to two orders of magnitude while only incurring a worst-case performance gap of 0.55–1.7 dB PSNR.

In this paper, we assume that the source and encoder dynamics are stationary and known (e.g. the rate-distortion behavior is known and statistically time-invariant). We comment on how these assumptions can be relaxed in future research, by incorporating online learning techniques into the proposed framework. We also discuss how the proposed work has applications to emerging Reconfigurable Video Coding MPEG standards. Although we focus on the interdependencies of the entire encoding pipeline in this paper, we envision that the proposed framework can be applied in an RVC encoder to decouple the decision processes of two or more codec component functions, by enabling them to autonomously select their optimal instantiations in response

to the behavior of their interconnected components.

ACKNOWLEDGMENT

The authors would like to thank Fangwen Fu for his valuable technical help throughout the writing of this paper.

APPENDIX A: BUFFER MODEL PROPERTIES

The IB buffer model introduced in Section III.A imposes a maximum average rate on each layer (i.e. $\bar{b}_l$) such that the bitstream Λ_l is guaranteed to comply with a maximum average rate (for example, under the IB buffer model, the bitstream Λ_l cannot exceed the maximum average channel rate $b_{\Lambda_l} \leq \sum_{k \in \Lambda_l} \bar{b}_k$). This is desirable when a receiver needs to subscribe to a bitstream Λ_l that is guaranteed to comply with its experienced channel rate. However, the IB buffer model is inflexible in its allocation of bits. For instance, in a two layer coder like the one illustrated in Fig. 2, if the base layer only uses an average of $\bar{b}_1 - z$ bits/DU, then the enhancement layer is still constrained to use only $\bar{b}_2$ bits/DU, even though it could use $\bar{b}_2 + z$ bits/DU without exceeding the maximum average rate $b_{\Lambda_2} \leq \bar{b}_1 + \bar{b}_2$. Despite its inflexibility, this buffer model is sufficient for optimizing the rate-distortion performance when the video source characteristics coupled with the layered coder satisfy the property

$$D\big(\bar{b}_1, \bar{b}_2\big) \leq D\big(\bar{b}_1 - z, \bar{b}_2 + z\big),\ z > 0\,, \tag{32}$$

which states that it is always better in terms of overall rate-distortion performance to allocate the maximum allowable amount of bits to the base layer. More generally, in an L layer setting, the property would imply that taking bits from an ancestor layer to allocate to a descendant layer would never improve the overall rate-distortion performance.

In contrast, the SBIC and SBJC buffer budget models introduced in Section III.B and Section III.C, respectively, can be used to optimize the rate-distortion performance when the sequence characteristics coupled with the layered coder do *not* satisfy (32). This can happen, for example, when using temporal scalability to encode a high-motion sequence [21].

APPENDIX B: BUFFER COSTS

In Section III, we introduced the concept of a buffer cost, which penalizes the layer as its buffer fills, thereby protecting against overflows that may result from a sudden burst in buffer arrivals. In this appendix we present illustrative buffer cost models for the IB and SBIC buffer models. Importantly, other forms of the

buffer cost models can be used, which may yield more conservative or more aggressive utilization of the buffer than the exemplary models used in this paper.

IB Buffer Cost: When layer l takes action a_l in state s_l, it incurs a *buffer cost*

$$g_l(s_l, a_l, f_l(\boldsymbol{a}(\Gamma_l))) = \left(\frac{1}{q_l^{\max}}\right)^2 \left(s_l + b_l(a_l, f_l(\boldsymbol{a}(\Gamma_l))) - \bar{b}_l\right)^2, \tag{33}$$

which is near its minimum when the buffer is empty and is maximized when the buffer is full. This buffer cost non-linearly penalizes the layer as its buffer fills.

SBIC Buffer Cost: If layer l is in state s_l, layer l takes action a_l, and layers $l' \in \Gamma_l$ take actions $\boldsymbol{a}(\Gamma_l)$, then layer l incurs a buffer cost

$$g_{\Lambda_l}(s_l, a_l, \boldsymbol{a}(\Gamma_l)) = \frac{1}{\left(q_{\Lambda_l}^{\max}\right)^2}\left(s_l + b_{\Lambda_l}(a_l, \boldsymbol{a}(\Gamma_l)) - \bar{b}_{\Lambda_l}\right)^2, \tag{34}$$

which is near its minimum when the buffer is empty and is maximized when the buffer is full.

APPENDIX C: CASE STUDY 3

When there are both data and budget dependencies, the cost function J takes a similar form to when there are only data dependencies; however, the buffer cost at layer l now depends directly on the actions $\boldsymbol{a}(\Gamma_l) = a_1, \ldots, a_{l-1}$ taken by its ancestor layers because they influence the number of bits arriving in the lth layer's buffer (see the buffer evolution equation for the SBIC buffer model defined in (6) and Fig. 5(b,c)). Hence, J can be additively decomposed as follows:

$$\begin{aligned} J(\boldsymbol{s}, \boldsymbol{a}) = {} & g_{\Lambda_1}(s_1, a_1) + \lambda_1 \cdot d_1(a_1) \\ & + \sum_{l=2}^{L} \left[g_{\Lambda_l}(s_l, a_1, \ldots, a_l) - \lambda_l \cdot \Delta d_l(a_l, f_l(a_1, \ldots, a_{l-1})) \right], \end{aligned} \tag{35}$$

where $g_{\Lambda_l}(s_l, a_1, \ldots, a_l)$ is defined as in (34); λ_1 weights the relative importance of the base layer's buffer cost and its distortion; and, λ_l, for $l \in \{2, \ldots, L\}$, weights the relative importance of each enhancement layer's buffer cost with its distortion reduction.

Using the factored transition probability function defined in (8) for the SBIC buffer model and the additive cost function defined in (35), the centralized VAL_I algorithm defined in (11) can be rewritten as

$$V^k(\boldsymbol{s}) = \min_{\boldsymbol{a} \in \boldsymbol{\mathcal{A}}} \left\{ \begin{array}{l} g_{\Lambda_1}(s_1, a_1) + \lambda_1 \cdot d_1(a_1) \\ + \sum_{l=2}^{L} \left[g_{\Lambda_l}(s_l, a_1, \ldots, a_l) - \lambda_l \cdot \Delta d_l(a_l, f_l(a_1, \ldots, a_{l-1})) \right] \\ + \gamma \sum_{\boldsymbol{s}' \in \boldsymbol{\mathcal{S}}} \prod_{l=1}^{L} p(s_l' \mid s_l, a_1, \ldots, a_l) V^{k-1}(\boldsymbol{s}') \end{array} \right\}. \tag{36}$$

We observe that the distortion reduction at layer $l \in \{2,\ldots,L\}$ is a function of $f_l(a_1,\ldots,a_{l-1})$, as in the case with only data dependencies. However, in this case, the buffer costs and transition probabilities at layer $l \in \{2,\ldots,L\}$ depend directly on the actions at the ancestor layers.

We may proceed in a similar fashion as we did in the case with only data dependences, and once again invoke decomposition principle 2 introduced in Section V.B. As before, we use the distortion $d_{\Lambda_{l-1}}(a_1,\ldots,a_{l-1})$ defined in (1) as a model of $f_l(a_1,\ldots,a_{l-1})$. In addition, we propose to use the rate $b_{\Lambda_{l-1}}(a_1,\ldots,a_{l-1})$ defined in (2) as a model for the actions $a_1,\ldots,a_{l-1}$. The rate is an ideal abstraction of $a_1,\ldots,a_{l-1}$ because it corresponds directly to the impact of the ancestor layers' actions on the descendent layer's buffer cost and buffer transition. Moreover, the rate can be easily inferred by any descendent layers by looking at the ancestor layer's buffer; or, alternatively, given any rate-distortion (distortion-rate) model (e.g. [22]), the descendent layers can infer the rate (distortion) given the distortion (rate) from the ancestor layers.

Because $b_{\Lambda_{l-1}}(a_1,\ldots,a_{l-1})$ is continuous, we quantize it to determine the *reference rate description* (RRD) $y_l = \psi_b\big(b_{\Lambda_{l-1}}(a_1,\ldots,a_{l-1})\big)$, where $\psi_b : \mathbb{R}_+ \mapsto \mathcal{Y}$ is a quantizer with reconstruction values in the set $\mathcal{Y}$. Using the RRD y_l and the RDD x_l, $b_{\Lambda_l}(a_1,\ldots,a_l)$ defined in (2) can be rewritten as

$$b_{\Lambda_l}(a_1,\ldots,a_l) = y_l + b_l(a_l, x_l), \tag{37}$$

which can be substituted into the SBIC model's buffer evolution equation defined in (6) and the buffer cost defined in (34). Subsequently, (36) can be approximated as

$$V^k(\boldsymbol{s}) \cong \min_{\boldsymbol{a}\in\boldsymbol{\mathcal{A}}} \left\{ \begin{array}{l} g_{\Lambda_1}(s_1,a_1) + \lambda_1 \cdot d_1(a_1) \\ +\sum_{l=2}^{L}\big[g_{\Lambda_l}(s_l,a_l,x_l,y_l) - \lambda_l \cdot \Delta d_l(a_l,x_l)\big] \\ +\gamma \sum_{\boldsymbol{s}'\in\boldsymbol{\mathcal{S}}} \prod_{l=1}^{L} p\big(s_l' \mid s_l,a_l,x_l,y_l\big) V^{k-1}\big(\boldsymbol{s}'\big) \end{array} \right\}. \tag{38}$$

Note that $g_{\Lambda_l}(s_l,a_l,x_l,y_l)$ and $p\big(s_l' \mid s_l,a_l,x_l,y_l\big)$ can be expressed more compactly as $g_{\Lambda_l}(\psi_l(s_l+y_l),a_l,x_l)$ and $p\big(s_l' \mid \psi_l(s_l+y_l),a_l,x_l,y_l\big)$, respectively, because y_l can be treated as an initial offset to the actual buffer state.

In the following, we describe an open-loop decomposition and a closed-loop decomposition for solving (38).

1) Open-loop decomposition

Similar to the open-loop decomposition in the case with only data dependencies, we begin with the base layer determining its locally optimal stationary policy $\pi_1^* : \mathcal{S}_1 \mapsto \mathcal{A}_1$ independently of the states and actions at

layers $l \in \{2,\ldots,L\}$. It does this using the local VAL_I defined in (23) to determine V_1 offline, which is then used to calculate π_1^* using (24).

Recall from (38) that the transition probability and cost functions at layer $l \in \{2,\ldots,L\}$ depend on the RDD x_l and RRD y_l from layer $l-1$. Hence, for layers $l \in \{2,\ldots,L\}$, we define a composite state $\vec{s}_l = (s_l, x_l, y_l)$. Based on its composite state, layer $l \in \{2,\ldots,L\}$ performs its local VAL_I offline as follows:

$$V_l^k(s_l,x_l,y_l) = \min_{a_l \in \mathcal{A}_l} \left\{ \begin{array}{l} g_{\Lambda_l}(s_l,a_l,x_l,y_l) - \lambda_l \cdot \Delta d_l(a_l,x_l) \\ +\gamma \sum_{s_l' \in \mathcal{S}_l} p(s_l' \mid s_l,a_l,x_l,y_l) \sum_{\substack{x_l' \in \mathcal{X} \\ y_l' \in \mathcal{Y}}} p(x_l',y') V_l^{k-1}(s_l',x_l',y_l') \end{array} \right\}, \tag{39}$$

where $p(x_l',y_l')$ is the stationary probability of the reference distortion and rate description pair $(x_l',y_l') \in \mathcal{X} \times \mathcal{Y}$ from layer $l-1$. The distribution $\{p(x_l,y_l) \mid x_l,y_l \in \mathcal{X} \times \mathcal{Y}\}$, for $l \in \{2,\ldots,L\}$, can be calculated by layer $l-1$ after it determines its optimal stationary policy π_{l-1}^*. Specifically,

$$p(x_l,y_l) = \sum_{s_{l-1} \in \mathcal{S}_{l-1}} \mu_{\pi_{l-1}^*}(s_{l-1}) I \left[\begin{array}{l} \psi_d\left(d_{\Lambda_{l-1}}(\pi_{l-1}^*(s_{l-1}))\right) = x_l \text{ and} \\ \psi_b\left(b_{\Lambda_{l-1}}(\pi_{l-1}^*(s_{l-1}))\right) = y_l \end{array} \right], \tag{40}$$

where $\mu_{\pi_{l-1}^*}(s_{l-1})$ is the stationary probability that layer $l-1$ is in state s_{l-1} given policy π_{l-1}^*, and $I[\cdot]$ is an indicator function. Hence, after layer $l-1$ performs its local VAL_I offline, it must calculate the distribution $\{p(x_l,y_l) \mid x_l,y_l \in \mathcal{X} \times \mathcal{Y}\}$ using (26) and forward it to layer l for use in its local VAL_I.

Finally, after computing its local state-value function V_l offline (obtained as $k \to \infty$ in (25)), layer $l \in \{2,\ldots,L\}$ can compute its locally optimal policy as

$$\pi_l^*(s_l,x_l,y_l) = \arg\min_{a_l \in \mathcal{A}_l} \left\{ \begin{array}{l} g_{\Lambda_l}(s_l,a_l,x_l,y_l) - \lambda_l \cdot \Delta d_l(a_l,x_l) \\ +\gamma \sum_{s_l' \in \mathcal{S}_l} p(s_l' \mid s_l,a_l,x_l,y_l) \sum_{\substack{x_l' \in \mathcal{X} \\ y_l' \in \mathcal{Y}}} p(x_l',y') V_l(s_l',x_l',y_l') \end{array} \right\}, \tag{41}$$

where $\pi_l^*(s_l,x_l,y_l)$ dictates the optimal action a_l^* to take in state s_l given the RDD x_l and the RRD y_l from layer $l-1$[10].

Table VIII summarizes the information requirements, memory complexity, and offline computational complexity associated with the open-loop decomposition with data and budget dependencies, and describes why it incurs a performance gap relative to the optimal centralized policy. This open-loop decomposition is similar to the one in Fig. 6, except that the message exchanged from layer $l-1$ to layer l is now

[10] Note that $V_l(s_l,x_l,y_l)$ and $\pi_l^*(s_l,x_l,y_l)$ can be represented more compactly as $V_l(\psi_l(s_l+y_l),x_l)$ and $\pi_l^*(\psi_l(s_l+y_l),x_l)$, respectively, because y_l can be treated as an initial offset to the actual buffer state.

$\{p(x_l, y_l) \mid x_l, y_l \in \mathcal{X} \times \mathcal{Y}\}$ and the information requirements are in Table VIII.

Table VIII. Decomposition information: Open-loop decomposition with data and budget dependencies.

	Transition probability function	Cost function
Information Requirements	$p(s_1' \mid s_1, a_1)$, $l = 1$ $p(s_l' \mid s_l, a_l, x_l, y_l)$, $p(x_l, y_l)$, o.w.	$g_{\Lambda_l}(s_1, a_1)$, $d_1(a_1)$, $l = 1$ $g_{\Lambda_l}(s_l, a_l, x_l, y_l)$, $\Delta d_l(a_l, x_l)$, o.w.
Memory Complexity	$\lvert\mathcal{S}_1\rvert^2 \lvert\mathcal{A}_1\rvert +$ $\sum_{l=2}^{L} \left(\lvert\mathcal{S}_l\rvert^2 \lvert\mathcal{A}_l\rvert \lvert\mathcal{X}_l\rvert \lvert\mathcal{Y}_l\rvert + \lvert\mathcal{X}_l\rvert \lvert\mathcal{Y}_l\rvert \right)$	$\lvert\mathcal{S}_1\rvert \lvert\mathcal{A}_1\rvert + \lvert\mathcal{A}_1\rvert +$ $\sum_{l=2}^{L} (\lvert\mathcal{S}_l\rvert \lvert\mathcal{A}_l\rvert \lvert\mathcal{X}_l\rvert \lvert\mathcal{Y}_l\rvert + \lvert\mathcal{A}_l\rvert \lvert\mathcal{X}_l\rvert)$
Computational Complexity	$O\left(\lvert\mathcal{S}_1\rvert^2 \lvert\mathcal{A}_1\rvert\right)$, per VAL_I at layer $l = 1$ $O\left(\lvert\mathcal{S}_l \times \mathcal{X}_l \times \mathcal{Y}_l\rvert^2 \lvert\mathcal{A}_l\rvert\right)$, per VAL_I o.w.	
Performance Gap	(i) Performance gap incurred because the descendent layers imperfectly model the ancestor layers based on the RDD and RRD. (ii) Performance gap incurred because the ancestor layers do not know their impact on the descendent layers.	

2) Closed-loop decomposition

As in the case with only data dependencies, the ancestor layers do not consider how their decisions impact the costs and state transitions at the descendant layers. As before, the solution to this problem is for the ancestor layers to maintain a model of the impact of their actions on the descendant layers' costs and state transitions, which can then be integrated into the ancestor layer's VAL_I algorithm to improve its decision policy.

The VAL_I algorithm and policy computation at the base layer are the same as they were in the closed-looped decomposition with only data dependencies (i.e. V_1^* is calculated offline using (28) and π_1^* is calculated using (29)). Subsequently, the closed-loop decomposition proceeds like the open-loop decomposition with data and budget dependencies: i.e., the base layer computes its joint RDD and RRD distribution $\{p(x_l, y_l) \mid x_l, y_l \in \mathcal{X} \times \mathcal{Y}\}$, and forwards it to the enhancement layer, which subsequently performs its own local VAL_I using (39).

The main difference between this closed-loop decomposition and the closed-looped decomposition in the case with only data dependencies is the precise definition of $M(s_2, a_1) = \left(J_M(s_2, a_1), p_M\left(s_2' \mid s_2, a_1\right)\right)$, which is used at the base layer to model its impact on the enhancement layer in accordance with decomposition principle 3 defined in Section V.B. Specifically, it is now necessary for $J_M(s_2, a_1)$ and $p_M\left(s_2' \mid s_2, a_1\right)$ to take into account the budget dependencies. By allowing the enhancement layer to share its cost function $g_2(s_2, a_2, x_2, y_2) - \lambda_2 \cdot \Delta d_2(a_2, x_2)$ and transition probability function $p(s_2' \mid s_2, a_2, x_2, y_2)$ with the base layer, the base layer can estimate its models by substituting the distortion description x_2 and rate

description y_2 with the actual expected distortion description $\psi_d(d_1(a_1))$ and rate description $\psi_b(b_1(a_1))$ given action a_1: i.e.,

$$J_M(s_2,a_1) = g_2(s_2,a_2,\psi_d(d_1(a_1)),\psi_b(b_1(a_1))) - \lambda_2 \cdot \Delta d_2(a_2,\psi_d(d_1(a_1)))$$
$$p_M(s_2' \mid s_2,a_1) = p(s_2' \mid s_2,a_2,\psi_d(d_1(a_1)),\psi_b(b_1(a_1))). \quad (42)$$

As in the closed-loop decomposition with only data dependencies, to decouple the interdependent decision processes, we must modify (42) to be independent of the enhancement layer's action. We assume that each action $a_2 \in \mathcal{A}_2$ is taken with probability $p_{\mathcal{A}_2}(a_2)$ (31).

Table V summarizes the information requirements, memory complexity, and offline computational complexity associated with the closed-loop decomposition with data and budget dependencies, and describes why it incurs a performance gap relative to the optimal centralized policy. This closed-loop decomposition is similar to the one in Fig. 7, except that, in order to account for the budget dependencies, the messages forwarded from the enhancement layer to the base layer are functions of the RRD y_2 in addition to the RDD x_2.

Table IX. Decomposition information: Closed-loop decomposition with data and budget dependencies.

	Transition probability function	Cost function
Information Requirements	$p(s_1' \mid s_1,a_1)$, $p_M(s_2' \mid s_2,a_1)$, $l=1$ $p(s_l' \mid s_l,a_l,x_l,y_l)$, $p(x_l,y_l)$, o.w.	$g_{\Lambda_l}(s_1,a_1)$, $d_1(a_1)$, $J_M(s_2,a_1)$, $l=1$ $g_{\Lambda_l}(s_l,a_l,x_l,y_l)$, $\Delta d_l(a_l,x_l)$, o.w.
Memory Complexity	$\lvert\mathcal{S}_1\rvert^2\lvert\mathcal{A}_1\rvert + \lvert\mathcal{S}_2\rvert^2\lvert\mathcal{A}_1\rvert +$ $\lvert\mathcal{S}_2\rvert^2\lvert\mathcal{A}_2\rvert\lvert\mathcal{X}_2\rvert\lvert\mathcal{Y}_2\rvert + \lvert\mathcal{X}_2\rvert\lvert\mathcal{Y}_2\rvert$	$\lvert\mathcal{S}_1\rvert\lvert\mathcal{A}_1\rvert + \lvert\mathcal{A}_1\rvert + \lvert\mathcal{S}_2\rvert\lvert\mathcal{A}_1\rvert$ $\lvert\mathcal{S}_2\rvert\lvert\mathcal{A}_2\rvert\lvert\mathcal{X}_2\rvert\lvert\mathcal{Y}_2\rvert + \lvert\mathcal{A}_2\rvert\lvert\mathcal{X}_2\rvert$
Computational Complexity	$O(\lvert\boldsymbol{\mathcal{S}}\rvert^2\lvert\mathcal{A}_1\rvert)$, per VAL_I at layer $l=1$ $O(\lvert\mathcal{S}_2\times\mathcal{X}_2\times\mathcal{Y}_2\rvert^2\lvert\mathcal{A}_2\rvert)$, at layer $l=2$	
Performance Gap	(i) Performance gap incurred because the descendent layers imperfectly model the ancestor layers based on the RDD and RRD. (ii) Performance gap incurred because the ancestor layers imperfectly model their impact on the descendent layers using $J_M(s_2,a_1)$ and $p_M(s_2' \mid s_2,a_1)$.	

REFERENCES

[1] A. Ortega, K. Ramchandran, and M. Vetterli, "Optimal trellis-based buffered compression and fast approximations," *IEEE Trans. on Image Processing*, vol. 3, no. 1, pp. 26-40, Jan. 1994.

[2] P. A. Chou, "Streaming media on demand," *Multimedia over IP and wireless Networks: compression, networking, and systems*, Ed. M. van der Schaar and P. Chou, Academic Press, Mar. 2007.

[3] J. Valentim, P. Nunes, and F. Pereira, "Evaluating MPEG-4 video decoding complexity for an alternative video complexity verifier model," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 12, no. 11, pp. 1034-1044, Nov. 2002.

[4] D. S. Turaga and T. Chen, "Hierarchical modeling of variable bit rate video sources," Packet Video Workshop, May 2001.

[5] V. K. Goyal, "Multiple description coding: compression meets the network," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 74-93, Sept. 2001.

[6] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Trans. on Multimedia*, vol. 8, pp. 390-404, 2001.

[7] W. Yang, F. Wu, Y. Lu, J. Cai, K. N. Ngan, and S. Li, "Scalable multiview video coding using wavelet,' *IEEE International Symposium on Circuits and Systems*, pp 6078-6081 Vol. 6, May 2005.

[8] M. Flierl and B. Girod, "Multivew video compression," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 66-76, Nov. 2007.

[9] ISO/IEC JTC1/SC29/WG11, "Reconfigurable video coding vision," Busan, Korea, Oct. 2008.

[10] J.-M. Hsiao and C.-J. Tsai, “Analysis of an SOC architecture for MPEG reconfigurable video coding framework,” *IEEE International Symposium on Circuits and Systems*, pp. 761-764, May 2007.
[11] B. Xie and W. Zeng, “Sequence-based rate control for constant quality video,” *IEEE International Conference on Image Processing*, Sept. 2002.
[12] M. M. Ghandi and M. Ghanbari, “A Lagrangian optimized rate control algorithm for the H.264/AVC Encoder,” *IEEE International Conference on Image Processing*, 2004.
[13] R. S. Sutton, and A. G. Barto, “Reinforcement learning: an introduction,” Cambridge, MA:MIT press, 1998.
[14] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: a survey,” *Journal of Artificial Intelligence Research 4*, pp. 237-285, May 2005.
[15] D. Wu, Y. T. Hou, and Y.-Q. Zhang, “Scalable video coding and transport over broadband wireless networks,” *Proc. of the IEEE*, vol. 89, no. 1, Jan. 2001.
[16] X. Guo, Y. Lu, F. Wu, W. Gao, and S. Li, “Distributed multi-view video coding,” *Proc. o The International Society for Optimal Engineering*, Jan. 2006.
[17] Y. Shoham and K. Leyton-Brown, *Multiagent systems: algorithmic, game theoretic and logical foundations*, Cambridge University Press, 2008.
[18] J. Apostolopoulos, “Path diversity and the use of multiple description coding,” *Multimedia over IP and wireless Networks: compression, networking, and systems*, Ed. M. van der Schaar and P. Chou, Academic Press, Mar. 2007.
[19] F. Fu and M. van der Schaar, “Decomposition Principles and Online Learning in Cross-Layer Optimization for Delay-Sensitive Applications”, submitted.
[20] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*, Prentice-Hall.
[21] M. van der Schaar and H. Radha, "A hybrid temporal-SNR fine-granular scalability for internet video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 318-331, Mar. 2001.
[22] K. Stuhlmüller, N. Färber, M. Link, and B. Girod, “Analysis of video transmission over lossy channels,” *IEEE J. Sel. Areas Commun.*, vol. 18, no. 6, Jun. 2000.