Branch and bound algorithms for rate-distortion optimized media streaming

Martin Röder, Jean Cardinal, Raouf Hamzaoui

Abstract— **We consider the problem of rate-distortion optimized streaming of packetized multimedia data over a single quality of service network using feedback and retransmissions. For a single data unit, we prove that the problem is NP-hard and provide efficient branch and bound algorithms that are much faster than the previously best solution based on dynamic programming. For a group of interdependent data units, we show how to compute optimal solutions with branch and bound algorithms. The branch and bound algorithms for a group of data units are much slower than the current state of the art, a heuristic technique known as sensitivity adaptation. However, in many real-world situations, they provide a significantly better rate-distortion performance.**

Keywords— **Streaming media, computational complexity, branch and bound algorithms.**

I. Introduction

A streaming media system allows the receiver to play back a compressed media bitstream continuously, while parts of it are still being received [1]. Chou and Miao [2] introduced a general framework for optimizing the rate-distortion performance of a streaming media system over a packet erasure channel. This framework enables a precise mathematical formulation of the problem of how and when to optimally transmit a group of interdependent data units before a delivery deadline. In a sender-driven scenario, each data unit can be sent to a client at a finite number of transmission opportunities. If the data unit is received before the delivery deadline, the client sends an acknowledgment packet to the server, which need not retransmit the data unit. Chou and Miao [2] showed that rate-distortion optimal transmission policies can be obtained by minimizing a Lagrangian. They first provided a dynamic programming algorithm to compute an optimal policy when a single data unit has to be transmitted. For the transmission of a group of interdependent data units, they proposed a heuristic iterative descent method, which uses the above dynamic programming algorithm to optimize for one data unit at a time, keeping the policies for the other units fixed. Data unit dependencies were modeled with a directed acyclic graph, which makes the system suitable to a wide range of source coders, including the new JVT video standard H.264/AVC [3]. The framework of Chou and Miao was adapted for a wireless channel [4], extended to include multiple arrival deadlines [5], and multiple streaming servers [6]. Chakareski and Girod [7] proposed a modification to the feedback sending mechanism. Instead of sending a separate acknowledgment packet for each received data unit, the receiver periodically sends a single feedback packet that informs the server on which data units were received when the acknowledgment packet was sent. Using an additive distortion model as an approximation to the distortion model of [2], Chakareski, Apostolopoulos, and Girod [8] provided a fast method for the computation of a transmission strategy for a group of data units. The simplified model allows a very efficient optimization at the cost of quality loss. The framework of Chou and Miao was also successfully used for networks that offer multiple qualities of service (QoS) [9].

Martin Röder and Raouf Hamzaoui are with the Department of Computer and Information Science, University of Konstanz, Konstanz, Germany, {roeder,hamzaoui}@inf.uni-konstanz.de. Jean Cardinal is with the Computer Science Department, Université Libre de Bruxelles, Brussels, Belgium, jcardin@ulb.ac.be. Parts of this paper were presented at DCC'04, Snowbird, Utah, March 2004.

In this paper, we focus on sender-driven transmission from a single server over a single-QoS network [2], but we also explain how the results can be extended to the receiver-driven case. We first show the NP-hardness of optimal rate-distortion streaming for a single data unit (Section III-A). In Section III-B, we provide a branch and bound algorithm to minimize the Lagrangian for a single data unit. Our algorithm is in practice much faster than the dynamic programming approach of Chou and Miao [2]. Moreover, in contrast to Chou and Miao who find only solutions whose error-cost points are on the lower convex hull of the admissible points, we provide another branch and bound algorithm that can compute optimal solutions that are unreachable with the Lagrange method. In Section IV, we show that for a policy of a group of data units to be optimal, the policy of each data unit must also be optimal. We exploit this result to give branch and bound algorithms for computing optimal policies for a group of data units. Section V explains how our branch and bound algorithms can be used with receiver-driven transmission. Section VI presents experimental results.

II. Terminology

In this section, we recall the terminology, notation, and assumptions needed to describe rate-distortion optimized media streaming in the context of sender-driven transmission over a single-QoS network [2]. A multimedia source (e.g., audio, image, video) is encoded and packetized into a finite set of *data units*. The interdependency between the data units is given by a directed acyclic graph (V, E), where the set of vertices V is the set of data units and the set of edges $E \subset V \times V$ is given by $(l', l) \in E$ if l' must be decoded so that l can also be decoded. For $(l', l) \in E$, we write $l' \prec l$ and say that l' is an *ancestor* of l. We also use the notation $l' \preceq l$ if $l' \prec l$ or $l' = l$. We denote by $B_l > 0$ the size of data unit l, $D_0 \geq 0$ the distortion if no data unit is decoded, and $\Delta D_l \geq 0$ the amount by which the distortion is decreased if data unit l is decoded compared to the distortion if only the ancestors of l are decoded. We assume that if the set $C \subset V$ of data units is decoded, then $D_0 - \sum_{l \in C} \Delta D_l$ is the resulting distortion. A channel packet containing a data unit and sent at time s can be either lost with probability ϵ_F, independent of s, or received at time s', where the *forward trip time* $FTT = s' - s$ is a random variable with probability density function p_F. Similarly, when a packet is sent from the client to the server through the feedback channel, it is either lost with probability ϵ_B or received after a *backward trip time* BTT, which is a random variable with probability density function p_B. We assume that each packet sent is lost or delayed independently of all other packets. We also define the *round trip time* RTT as the sum $FTT + BTT$. For convenience, by setting $FTT = +\infty$ when a data unit is lost, one can extend the random variable FTT to make it include packet loss [2]. In this situation, the cumulative distribution function of the extended random variable, $\overline{FTT}$ is

$$P\{\overline{FTT} \leq \tau\} = \int_0^\tau (1 - \epsilon_F) p_F(t) dt.$$

The same extension can be done for BTT, yielding an extended random variable, $\overline{BTT}$. Finally, $\overline{RTT}$ will denote the sum $\overline{FTT} + \overline{BTT}$.

In a sender-driven context, a data unit can be sent to the receiver at time opportunities $s_0, \ldots, s_{N-1}$ with $s_0 < \ldots < s_{N-1} < s_{DTS}$, where s_{DTS} is a delivery deadline. A transmission policy $\pi = (\pi(0), \pi(1), \ldots, \pi(N-1)) \in \{0,1\}^N$ of length $\text{len}(\pi) = N$ is used to describe the transmission of a data unit. Here $\pi(i) = 0$ means that the data unit should not be sent at opportunity s_i, whereas $\pi(i) = 1$ means that the data unit should be sent at opportunity s_i if no acknowledgment packet was received from the feedback channel before s_i, and that the data unit should not be sent at this opportunity otherwise.

We define an error $\epsilon(\pi)$ for policy π as the probability that the data unit does not reach its destination before time s_{DTS}, that is,

$$\epsilon(\pi) = \prod_{i:\pi(i)=1} P\{\overline{FTT} > s_{DTS} - s_i\}. \quad (1)$$

We also define a cost $\rho(\pi)$ for policy π as the expected number of data unit transmissions. One can show (see Appendix 1 in [10]) that

$$\rho(\pi) = \sum_{i:\pi(i)=1} \left(\prod_{j<i:\pi(j)=1} P\{\overline{RTT} > s_i - s_j\} \right). \quad (2)$$

The first problem discussed in this paper is the computation of rate-distortion optimal policies for the transmission of a data unit. We say that a policy π^* is optimal if there exists no policy π such that $\epsilon(\pi) \leq \epsilon(\pi^*)$ and $\rho(\pi) < \rho(\pi^*)$. Note that a policy π^* is optimal if and only if there exists $\epsilon > 0$ such that π^* minimizes $\rho(\pi)$ subject to the constraint $\epsilon(\pi) \leq \epsilon$. Chou and Miao [2] claim that computing the set of optimal policies is infeasible and propose to determine instead only optimal policies π for which the cost-error points $(\rho(\pi), \epsilon(\pi))$ are on the lower convex hull [11] of the set $\{(\rho(\pi), \epsilon(\pi)),\ \pi \in \{0,1\}^N\}$. We call these policies *convex hull* policies. A policy π^* is a convex hull policy if and only if there exists $\lambda \geq 0$ such that π^* minimizes the Lagrangian

$$J_\lambda(\pi) = \epsilon(\pi) + \lambda \rho(\pi). \quad (3)$$

For a given λ, brute force minimization of the Lagrangian requires $O(N^2 2^N)$ time. Chou and Miao [2] reduce the complexity of the minimization problem to $O(N 2^N)$ with dynamic programming.

We also consider the problem of rate-distortion optimal transmission of a group of interdependent data units. The transmission policies for the group of data units is described by a policy vector $\vec{\pi} = (\pi_1, \ldots, \pi_L)$, where π_i, $i \in \{1, \ldots, L\}$ is the transmission policy for the ith data unit of the group. The expected transmission cost (the expected rate) for $\vec{\pi}$ is

$$R(\vec{\pi}) = \sum_{l=1}^{L} B_l \rho(\pi_l). \quad (4)$$

By assuming independence of the transmission processes, the expected distortion for $\vec{\pi}$ is

$$D(\vec{\pi}) = D_0 - \sum_{l=1}^{L} \Delta D_l \prod_{l' \preceq l} (1 - \epsilon(\pi_{l'})). \quad (5)$$

A policy vector $\vec{\pi}^*$ is optimal if there exists no policy vector $\vec{\pi}$ such that $D(\vec{\pi}) \leq D(\vec{\pi}^*)$ and $R(\vec{\pi}) < R(\vec{\pi}^*)$. Here also Chou and Miao [2] propose to compute only optimal policy vectors $\vec{\pi}$ for which the rate-distortion points

$(R(\vec{\pi}), D(\vec{\pi}))$ are on the lower convex hull of the set $\{(R(\vec{\pi}), D(\vec{\pi})),\ \vec{\pi} \in (\{0,1\}^N)^L\}$. We call these policy vectors convex hull policy vectors. A policy vector $\vec{\pi}^*$ is a convex hull policy vector if and only if there exists $\lambda \geq 0$ such that $\vec{\pi}^*$ minimizes the Lagrangian

$$J_\lambda(\vec{\pi}) = D(\vec{\pi}) + \lambda R(\vec{\pi}). \tag{6}$$

To minimize J_λ for a given λ, Chou and Miao [2] apply a heuristic iterative descent algorithm, which they call sensitivity adaptation (SA). The idea is to minimize $J_\lambda(\pi_1, \ldots, \pi_L)$ one policy at a time, keeping the other policies fixed. Starting from an initial policy vector $\vec{\pi}^{(0)} = (\pi_1^{(0)}, \ldots, \pi_L^{(0)})$, where $\pi_1^{(0)} = \cdots = \pi_L^{(0)} = (1, \ldots, 1)$, a sequence of policy vectors $\vec{\pi}^{(k)} = (\pi_1^{(k)}, \ldots, \pi_L^{(k)})$, $k = 1, 2, \ldots$, is constructed as follows. Select $l_k \in \{1, \ldots, L\}$. For $i \neq l_k$, set $\pi_i^{(k)} = \pi_i^{(k-1)}$ and for $i = l_k$, let

$$\begin{aligned} \pi_i^{(k)} &= \arg\min_{\pi_i} J_\lambda(\pi_1^{(k)}, \ldots, \pi_{i-1}^{(k)}, \pi_i, \pi_{i+1}^{(k)}, \ldots, \pi_L^{(k)}) \\ &= \arg\min_{\pi_i} S_i^{(k)} \epsilon(\pi_i) + \lambda B_i \rho(\pi_i), \end{aligned} \tag{7}$$

where

$$S_i^{(k)} = \sum_{l', i \preceq l'} \Delta D_{l'} \prod_{l'' \prec l'} (1 - \epsilon(\pi_{l''}^{(k)})).$$

Since the sequence $J_\lambda(\vec{\pi}^{(k)})$ is nonincreasing and bounded below by zero, it converges to a local minimum. When the previous dynamic programming algorithm is used to solve (7), the time complexity of the SA algorithm is $O(nN2^N)$, where n is the number of iterations needed for convergence.

III. Single data unit

A. NP-hardness

We show that finding the best transmission policy for a single data unit is NP-hard by using a reduction from the knapsack problem [12]. We first formally define the problem Policy.

Definition 1 (Policy) An instance of problem Policy consists of a positive integer N, $N+1$ real numbers $s_0 \leq s_1 \leq \ldots \leq s_{N-1} \leq s_{DTS}$, two real random variables FTT and RTT with $FTT \leq RTT$, and a real number $\epsilon^* \in [0,1]$. An optimal solution of the problem is a policy $\pi \in \{0,1\}^N$ satisfying $\epsilon(\pi) \leq \epsilon^*$ and minimizing the cost $\rho(\pi)$.

For simplicity, and without loss of generality, we assume in Definition 1 that $\overline{FTT} = FTT$ and $\overline{RTT} = RTT$, that is, the loss probabilities ϵ_F and ϵ_B are equal to zero. Note that although we included the random variables in the definition, the problem is essentially combinatorial: the finite set of probabilities $P\{FTT > s_{DTS} - s_i\}$ and $P\{RTT > s_i - s_j\}$ is the only information we need about FTT and RTT.

Lemma 1: If a given instance of problem Policy has an optimal solution $\pi^* \neq (0, \ldots, 0)$, then there exists an optimal solution π for the same instance with $\pi(0) = 1$.

Proof: Suppose that $\pi^*(0) = 0$. Let π be a policy identical to π^*, except that $\pi(0) = 1$, and the first element of π^* equal to one is set to zero in π. Let k be the index of this element. We have $\rho(\pi) \leq \rho(\pi^*)$, since the probability $P\{RTT > s_i - s_k\}$ is replaced by the probability $P\{RTT > s_i - s_0\}$, which is never greater. We also have $\epsilon(\pi) \leq \epsilon(\pi^*)$, because $P\{FTT > s_{DTS} - s_0\} \leq P\{FTT > s_{DTS} - s_k\}$. Hence π is optimal too. ■

Definition 2 (Knapsack) An instance of problem Knapsack consists of a positive integer m, positive real values w_i and v_i, $i = 1, \ldots, m$, and a nonnegative real number B. An optimal solution of Knapsack is a set $S \subseteq \{1, 2, \ldots, m\}$ satisfying $\sum_{i \in S} w_i \geq B$ and minimizing $\sum_{i \in S} v_i$.

Theorem 1: Policy is NP-hard .

Proof: We prove the result by constructing a polynomial transformation from Knapsack into Policy. Without loss of generality, we consider a restricted version of Knapsack with the following additional assumptions: the values v_i and w_i are decreasing with respect to i, and $v_i \in [0,1]$ for all i. It is not difficult to show that Knapsack remains NP-hard under these restrictions [12], [13].

Let $(m, \{v_i\}, \{w_i\}, B)$ be such an instance of Knapsack. The corresponding instance of Policy is defined by $N = m + 1$, $s_0 = -\delta$ for some value $\delta > 1$, $s_i = \frac{i}{N}$, $i = 1, 2, \ldots, N-1$, $s_{DTS} = 1$, two random variables FTT and RTT whose cumulative distribution functions $P\{FTT \leq x\}$ and $P\{RTT \leq x\}$ satisfy

$$P\{FTT \leq 1 - \frac{i}{N}\} = 1 - 2^{-w_i}, \tag{8}$$

$$P\{FTT \leq \delta + 1\} = 1 - \mu, \tag{9}$$

$$P\{RTT \leq \delta\} = 0, \tag{10}$$

$$P\{RTT \leq \delta + \frac{i}{N}\} = 1 - v_i \tag{11}$$

for $i = 1, 2, \ldots, N-1$ and some value μ such that $0 < \mu < \min\{2^{-w_1}, v_{N-1}, 2^B\}$. The values $P\{FTT \leq x\}$ and $P\{RTT \leq x\}$ that are not defined above can be chosen by any kind of continuous nondecreasing interpolation between the defined values. We also let their limits as x tends to infinity and to zero be equal to one and zero, respectively. This is illustrated in Fig. 1. One can show that the two cumulative distribution functions defined in this way are well-behaved nondecreasing functions taking values in $[0,1]$. In particular, since the w_i are sorted in decreasing order, the values of $P\{FTT \leq x\}$ from (8) are increasing with x. Since μ is chosen so that $\mu \leq 2^{-w_1}$ this is true with respect to (9), too. The cumulative distribution function $P\{RTT \leq x\}$ satisfying

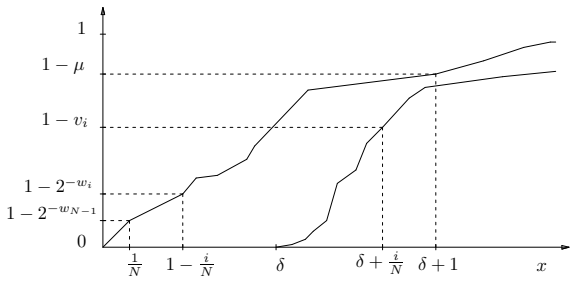

Fig. 1. Illustration of the distributions used in the proof. The top curve is $P\{FTT \leq x\}$, and the bottom curve is $P\{RTT \leq x\}$.

(10)-(11) is also increasing if the v_i are sorted in decreasing order. Next, we must check that given these two cumulative distribution functions, there exist two random variables FTT and RTT such that $FTT \leq RTT$. From the stochastic order relation ([14], Theorem A, p. 6), a sufficient condition is $P\{FTT \leq x\} \geq P\{RTT \leq x\}$. This can be ensured by setting:

$$\begin{aligned} P\{FTT \leq \delta+1\} &\geq P\{RTT \leq \delta + \frac{i}{N}\}, \\ P\{FTT \leq \delta+1\} &\geq P\{RTT \leq \delta + \frac{N-1}{N}\}, \\ 1-\mu &\geq 1-v_{N-1}, \\ \mu &\leq v_{N-1}, \end{aligned}$$

which is guaranteed by the choice of μ. Finally, we set $\epsilon^* = \frac{\mu}{2^B}$, which belongs to $[0,1]$.

Now we show that this instance of POLICY is equivalent to the original instance of KNAPSACK. Let $\pi \neq (0,\ldots,0)$ be an optimal solution for POLICY. Using Lemma 1, we can assume that $\pi(0)=1$. From (9), we get

$$\epsilon(\pi) = \mu \prod_{i>0:\pi(i)=1} \left(1 - P\{FTT \leq 1 - \frac{i}{N}\}\right).$$

So $\epsilon(\pi) \leq \epsilon^*$ is equivalent to the inequalities

$$\begin{aligned} \mu \prod_{i>0:\pi(i)=1} 2^{-w_i} &\leq \frac{\mu}{2^B} \\ \sum_{i>0:\pi(i)=1} w_i &\geq B. \end{aligned} \quad (12)$$

Moreover, from (10)-(11), we have

$$P\{RTT > s_i - s_j\} = \begin{cases} v_i \text{ if } j=0 \\ 1 \text{ otherwise.} \end{cases}$$

Thus

$$\begin{aligned} \rho(\pi) &= \sum_{i:\pi(i)=1} \left(\prod_{j<i:\pi(j)=1} P\{RTT > s_i - s_j\} \right) \\ &= 1 + \sum_{i>0:\pi(i)=1} v_i. \end{aligned} \quad (13)$$

Because of (12) and (13), π will yield an optimal solution S of KNAPSACK simply by letting $S = \{i > 0 : \pi(i) = 1\}$. Conversely, an optimal solution S of KNAPSACK yields an optimal solution π for POLICY. Hence POLICY is at least as hard as KNAPSACK. ■

B. Branch and bound algorithms

In this section, we propose a class of branch and bound algorithms to compute convex hull and optimal policies. The algorithms exploit the following important bounds.

Suppose that a policy π of length N begins with prefix π'. That is, $\pi'(i) = \pi(i)$ for $i = 0,\ldots,\text{len}(\pi')-1$ and $\text{len}(\pi') \leq N$. Since $\epsilon(\pi)$ is smallest if we transmit as often as possible, a lower bound for $\epsilon(\pi)$ is

$$\epsilon_{\min}(\pi') = \epsilon((\pi'(0),\ldots,\pi'(\text{len}(\pi')-1), \underbrace{1,\ldots,1}_{N-\text{len}(\pi')}).$$

On the other hand, $\rho(\pi)$ is smallest if we do not transmit at all, thus a lower bound for $\rho(\pi)$ is

$$\rho_{\min}(\pi') = \rho((\pi'(0),\ldots,\pi'(\text{len}(\pi')-1), \underbrace{0,\ldots,0}_{N-\text{len}(\pi')}).$$

A combination of $\epsilon_{\min}(\pi')$ and $\rho_{\min}(\pi')$ gives a lower bound for the expected Lagrangian $J_\lambda(\pi)$ of a policy π with prefix π' as $J_{\lambda,\min}(\pi') = \epsilon_{\min}(\pi') + \lambda\rho_{\min}(\pi')$.

B.1 Convex hull policies

We first propose a branch and bound algorithm to compute a convex hull policy for a fixed Lagrange multiplier λ. A policy prefix of size n is recursively extended to policy prefixes of size $n+1$. The recursive calls are stopped as soon as the lower bound $J_{\lambda,\min}$ exceeds $J_\lambda(\pi)$, where π is the currently best transmission policy, or the size of a policy prefix reaches N. In the latter case, π is updated with the policy prefix of size N.

More precisely, we initialize a policy π with an approximation of the optimal policy. This could be the solution of a previous optimization with similar parameters, or an arbitrary policy, e.g. $(0,\ldots,0)$. The recursive part of the algorithm starts with an empty policy prefix π'. A single recursion on a policy prefix π' is as follows. We determine all possible policy prefixes π'_k whose length is the length of π' increased by 1. There are only two such policy prefixes (one can append either 0 or 1 to π'). Let π'_0 be π' with 0 appended, and π'_1 be π' with 1 appended. We then calculate $J_{\lambda,\min}(\pi'_k)$ for $k=0,1$. If $J_{\lambda,\min}(\pi'_k) > J_\lambda(\pi)$, we know that no policy with prefix π'_k can be better than π. Otherwise, if $\text{len}(\pi'_k) = N$, we found a better policy than π; so we set $\pi = \pi'_k$. If $\text{len}(\pi'_k) < N$, the described

recursion is done again on π'_k. If the recursion has to be done on both π'_0 and π'_1, it should be done in increasing $J_{\lambda,\min}(\pi'_k)$ order. When the algorithm ends, π will be the optimal policy. We denote this algorithm by LBB. A pseudo-code of LBB is given in [10], Appendix 2.

It is straightforward to update the error bound $\epsilon_{\min}$ at each recursive step in constant time, but for the cost bound $\rho_{\min}$, the time complexity of an update is linear in N. The worst-case complexity of the algorithm is therefore $O(N2^N)$, since the number of recursive calls is bounded by $O(2^N)$. However, in practice, the pruning allows a substantial speed-up.

We now give a branch and bound algorithm to compute all convex hull policies in a single run. The main task will be to test if the cost-error point of a policy with a given prefix may belong to Γ, the lower convex hull of the points already checked, and to update Γ with insertions and deletions. For simplicity, we will identify a policy with the cost-error point associated to it.

We assume that Γ is sorted in nondecreasing cost and nonincreasing error order. This can be easily done, e.g., with a binary search tree. We denote the ith policy in Γ by $\Gamma(i)$; thus $\rho(\Gamma(i)) \leq \rho(\Gamma(i+1))$ and $\epsilon(\Gamma(i)) \geq \epsilon(\Gamma(i+1))$, $i = 0, \ldots, |\Gamma| - 2$. Then, we define a function *CHTest(*Γ*,*π'*)* that tests if a policy with prefix π' can belong to the lower convex hull Γ. The test consists of the following steps.

1. Try to find $k \in \{0, \ldots, |\Gamma| - 2\}$ such that $\rho(\Gamma(k)) < \rho_{\min}(\pi') \leq \rho(\Gamma(k+1))$.
2. If no such k exists, set *CHTest(*Γ*,*π'*)* = "yes".
3. If such a k exists, set *CHTest(*Γ*,*π'*)* to "no" if $(\rho_{\min}(\pi'), \epsilon_{\min}(\pi'))$ is above the line joining the points $\Gamma(k)$ and $\Gamma(k+1)$, and set it to "yes", otherwise.

To insert a new policy into Γ, we define a procedure *CHInsert(*Γ*,*π*)*. We assume that *CHTest(*Γ*,*π*)* = "yes" at the time we call *CHInsert(*Γ*,*π*)*. The procedure first inserts π into Γ and then convexifies Γ by ensuring that the lines joining neighboring points have an increasing slope. This can be done by checking if the left neighbor of the inserted point is above the line joining the left neighbor of the left neighbor of the inserted point to the inserted point. If it is, the left neighbor of the inserted point is removed and the check is repeated. The same check is done on the right neighbors of the inserted point.

Figure 2 illustrates *CHTest* and *CHInsert* on an example. Suppose that Γ currently consists of $\Gamma(0), \ldots, \Gamma(5)$ and that we want to test if a policy with prefix π'_1 can belong to the lower convex hull. *CHTest(*Γ*,*π'_1*)* equals "no", because $(\rho_{\min}(\pi'_1), \epsilon_{\min}(\pi'_1))$ is above the line joining $\Gamma(2)$ and $\Gamma(3)$. Since all policies with prefix π'_1 are in the shaded area, none of them is in the lower convex hull. Suppose we want to insert π_2 into Γ.

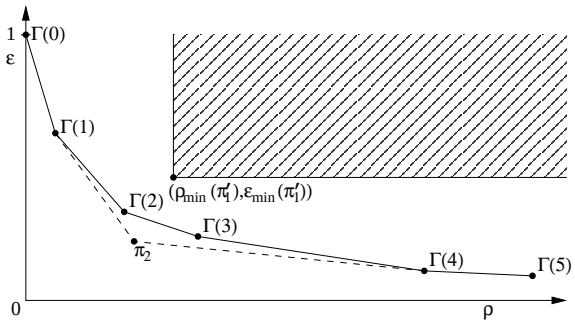

Fig. 2. Example of convex hull test and insertion

CHInsert(Γ*,*π_2*)* removes $\Gamma(2)$ and $\Gamma(3)$ from Γ, because they are above the lines connecting $\Gamma(1)$ and π_2 and π_2 and $\Gamma(4)$, respectively.

The branch and bound method to compute all convex hull policies is similar to the branch and bound method for the Lagrange optimization. We initialize Γ with $\Gamma = \{(0, \ldots, 0), (1, \ldots, 1)\}$. We then start the recursive part of the algorithm with prefix $\pi' = ()$. In each single recursion, we determine π'_k, $k = 0, 1$ as in the Lagrange optimization. For each π'_k, we then determine *CHTest(*Γ*,*π'_k*)*, and if it equals "yes", we either insert π'_k into Γ with *CHInsert(*Γ*,*π'_k*)* if $\text{len}(\pi'_k) = N$, or we do the recursion again on π'_k if $\text{len}(\pi'_k) < N$.

When the algorithm stops, Γ is the set of convex hull policies with length N. A pseudo-code of the algorithm is given in [10], Appendix 3. Note that by exploiting Lemma 1, the algorithm can be accelerated by initializing π' with (1). In this case, Γ may not contain all convex hull policies, but all points $(\rho(\pi), \epsilon(\pi))$ such that π is a convex hull policy are found.

B.2 Optimal policies

The branch and bound approach to compute a convex hull policy for a fixed Lagrange multiplier can also be adapted to find an optimal policy for a given maximum cost $\rho_{\max}$. In each recursion, we discard all policies with prefix π' if $\epsilon_{\min}(\pi') > \epsilon(\pi)$ or $\rho_{\min}(\pi') > \rho_{\max}$ or $\text{len}(\pi') = N$, $\epsilon_{\min}(\pi') = \epsilon(\pi)$, and $\rho_{\min}(\pi') > \rho(\pi)$. Here π denotes the current best policy. We denote this algorithm by CBB. A pseudo-code of CBB is given in [10], Appendix 4.

The branch and bound algorithm to compute all convex hull policies can be adapted to find all optimal policies. We now use a test function *OPTest(*Π*,*π'*)* that checks if policies with prefix π' can be optimal among the set $\Pi = \{\Pi(0), \ldots, \Pi(|\Pi| - 1)\}$ of all current optimal policies. Here we assume that Π is sorted in nondecreasing cost and nonincreasing error order. The test is done as follows.

1. Try to find $k \in \{0, \ldots, |\Pi| - 2\}$ such that $\rho(\Pi(k)) < \rho_{\min}(\pi') \leq \rho(\Pi(k+1))$.

2. If no such k exists, $\rho_{\min}(\pi')$ is either smaller or equal than all $\rho(\Pi(k))$, or larger than all $\rho(\Pi(k))$. If it is smaller or equal, set *OPTest(*Π,π'*)*="yes". If it is larger, set it to "yes" if $\epsilon_{\min}(\pi') < \epsilon(\Pi(|\Pi|-1))$ and to "no" otherwise.
3. If such a k exists, set *OPTest(*Π,π'*)* to "yes" if $\epsilon_{\min}(\pi') < \epsilon(\Pi(k))$ and set it to "no", otherwise.

To insert a new policy π into the current set of optimal policies Π, we use a procedure *OPInsert(*Π,π*)*. Here we also assume that *OPTest(*Π,π*)*= "yes" when calling *OPInsert(*Π,π*)*. The procedure tries to find the smallest $k \in \{0,\ldots,|\Pi|-1\}$ such that $\rho(\pi) \leq \rho(\Pi(k))$. If no such k exists, it sets $k = |\Pi|$. If $k = 0$, π is inserted at the beginning of Π, if $k = |\Pi|$, π is inserted at the end of Π, otherwise π is inserted between $\Pi(k-1)$ and $\Pi(k)$. Note that after the insertion, we have $\pi = \Pi(k)$. Finally, all $\Pi(i)$ with $i > k$ and $\epsilon(\Pi(i)) \geq \epsilon(\pi)$ are removed from Π.

The algorithm for computing all optimal policies starts by initializing Π with $\{(0,\ldots,0),(1,\ldots,1)\}$ and π' with (). A pseudo-code of the algorithm is given in [10], Appendix 5. Again, by exploiting Lemma 1, the algorithm can be accelerated by initializing π' with (1).

In the computation of both Γ and Π, the worst-case complexity of a single insertion in the set Γ (resp. Π) is linear in the size of the set. We can nevertheless prove a worst-case complexity of $O(N2^N)$ in the two cases by remarking that no more than 2^N points can be removed from Γ (resp. Π) in a single run of the algorithm, and that the complexity of a deletion is $O(\log|\Gamma|) = O(N)$ (resp. $O(\log|\Pi|) = O(N)$). This is a worst-case complexity if we assume that the policies are stored in a balanced binary tree. Again, in practice, the pruning procedures *CHTest* and *OPTest* should yield substantial acceleration ratios.

IV. Group of data units

We study in this section the complexity of rate-distortion optimized streaming for a group of interdependent data units. Since the rate-distortion optimization problem is NP-hard for a single data unit, it is also NP-hard for a group of data units. We now provide two important results that give relationships between optimal (resp. convex hull) policy vectors and optimal (resp. convex hull) policies.

Lemma 2: If $\vec{\pi}^* = (\pi_1^*,\ldots,\pi_L^*)$ is an optimal policy vector, then all policies $\pi_1^*,\ldots,\pi_L^*$ are optimal.

Proof: We prove the result by contradiction. Let $\vec{\pi} = (\pi_1,\ldots,\pi_L)$ be a policy vector with at least one non-optimal policy. Let, without loss of generality, π_L be a non-optimal policy, that is, there exists a policy π' such that $\epsilon(\pi') \leq \epsilon(\pi_L)$ and $\rho(\pi') < \rho(\pi_L)$. Let $\vec{\pi}' = (\pi_1',\ldots,\pi_L')$ with $\pi_i' = \pi_i,\ i = 1,\ldots,L-1$ and $\pi_L' = \pi'$. Then

$$D(\vec{\pi}) \geq D_0 - \sum_{l=1}^{L} \Delta D_l \prod_{l' \preceq l} (1-\epsilon(\pi_{l'}')) = D(\vec{\pi}')$$

and

$$R(\vec{\pi}) > \sum_{l=1}^{L} B_l \rho(\pi_l') = R(\vec{\pi}').$$

Thus, $\vec{\pi}$ cannot be optimal. ■

We have a similar result for convex hull policy vectors.

Lemma 3: If $\vec{\pi}^* = (\pi_1^*,\ldots,\pi_L^*)$ is a convex hull policy vector that minimizes the Lagrangian J_λ for a $\lambda > 0$, then all policies $\pi_1^*,\ldots,\pi_L^*$ are convex hull policies.

Proof: We prove the result by contradiction. Assume that $\vec{\pi} = (\pi_1,\ldots,\pi_L)$ is a convex hull policy vector that minimizes J_λ for a $\lambda > 0$. Suppose that at least one policy in $\vec{\pi}$ is not a convex hull policy. Let, without loss of generality, π_L be one such policy, that is, for each $\lambda' \geq 0$, there exists a policy π' such that $J_{\lambda'}(\pi') < J_{\lambda'}(\pi_L)$. Let $A_l(\vec{\pi}) = \prod_{l' \preceq l}(1-\epsilon(\pi_{l'}))$ and $A_l'(\vec{\pi}) = \prod_{l' \prec l}(1-\epsilon(\pi_{l'}))$. Then for all $\lambda > 0$

$$\begin{aligned} J_\lambda(\vec{\pi}) &= D(\vec{\pi}) + \lambda R(\vec{\pi}) \\ &= D_0 + \sum_{l=1}^{L} [-\Delta D_l A_l(\vec{\pi}) + \lambda B_l \rho(\pi_l)] \\ &= D_0 + \sum_{l=1}^{L-1} [-\Delta D_l A_l(\vec{\pi}) + \lambda B_l \rho(\pi_l)] \\ &\quad -\Delta D_L A_L'(\vec{\pi}) + \Delta D_L \epsilon(\pi_L) A_L'(\vec{\pi}) \\ &\quad +\lambda B_L \rho(\pi_L). \end{aligned}$$

If $\Delta D_L > 0$ and $\epsilon(\pi_{l'}) < 1$ for all $l' \prec L$, then with $\lambda' = \frac{\lambda B_L}{\Delta D_L A_L'(\vec{\pi})}$ and π' such that $J_{\lambda'}(\pi') < J_{\lambda'}(\pi_L)$, we have

$$\begin{aligned} J_\lambda(\vec{\pi}) &= D_0 + \sum_{l=1}^{L-1} [-\Delta D_l A_l(\vec{\pi}) + \lambda B_l \rho(\pi_l)] \\ &\quad -\Delta D_L A_L'(\vec{\pi}) + \Delta D_L J_{\lambda'}(\pi_L) A_L'(\vec{\pi}) \\ &> D_0 + \sum_{l=1}^{L-1} [-\Delta D_l A_l(\vec{\pi}) + \lambda B_l \rho(\pi_l)] \\ &\quad -\Delta D_L A_L'(\vec{\pi}) + \Delta D_L J_{\lambda'}(\pi') A_L'(\vec{\pi}) \\ &= J_\lambda(\vec{\pi}'), \end{aligned}$$

where $\vec{\pi}' = (\pi_1,\ldots,\pi_{L-1},\pi')$.

If $\Delta D_L = 0$ or if there exists $l' \prec L$ with $\epsilon(\pi_{l'}) = 1$, then

$$\begin{aligned} J_\lambda(\vec{\pi}) &= D_0 + \sum_{l=1}^{L-1} [-\Delta D_l A_l(\vec{\pi}) + \lambda B_l \rho(\pi_l)] \\ &\quad +\lambda B_L \rho(\pi_L). \end{aligned}$$

Since the convex hull policy $\pi' = (0, \ldots, 0)$ is the only policy whose cost is zero, and since π_L is not a convex hull policy, we have $\rho(\pi_L) > 0$. Thus,

$$\begin{aligned} J_\lambda(\vec{\pi}) &> D_0 + \sum_{l=1}^{L-1} [-\Delta D_l A_l(\vec{\pi}) + \lambda B_l \rho(\pi_l)] \\ &= J_\lambda(\vec{\pi}'), \end{aligned}$$

where $\vec{\pi}' = (\pi_1, \ldots, \pi_{L-1}, \pi')$. So also in this case we have a contradiction to our initial assumption. ■

Note that the convex hull policy $\pi = (1, \ldots, 1)$ gives a convex hull policy vector $(\pi, \ldots, \pi)$ that minimizes J_λ for $\lambda = 0$.

Lemma 2 (resp. Lemma 3) allows us to break the problem of finding optimal policy vectors (resp. convex hull policy vectors) for a group of data units into two parts. First, use the branch and bound algorithm of Section III-B.2 (resp. Section III-B.1) to compute the set Π (resp. Γ) of all optimal (resp. convex hull) policies for a single data unit. Then use another branch and bound algorithm, which is similar to the first one, except that instead of having the choice between two branches at each step, we have to choose the next policy among the set Π (resp. Γ). We can derive lower bounds for pruning the recursion tree as in the single policy case.

As in the single data unit case, if we know that policy vector $\vec{\pi}$ begins with prefix $\vec{\pi}' = (\pi'_1, \ldots, \pi'_{\text{len}(\vec{\pi}')})$, $\pi'_i = \pi_i$ for $i = 1, \ldots, \text{len}(\vec{\pi}')$, $\text{len}(\vec{\pi}') \leq L$, we can give bounds for $D(\vec{\pi})$ and $R(\vec{\pi})$. Let $\mathbf{0} = (0, \ldots, 0)$ and $\mathbf{1} = (1, \ldots, 1)$. As $D(\vec{\pi})$ is smallest if we transmit with the smallest error for each data unit, a lower bound for $D(\vec{\pi})$ is

$$D_{\min}(\vec{\pi}') = D(\pi'_1, \ldots, \pi'_{\text{len}(\vec{\pi}')}, \underbrace{\mathbf{1}, \ldots, \mathbf{1}}_{L - \text{len}(\vec{\pi}')}) \qquad (14)$$

since $\mathbf{1}$ is a policy with the lowest error. On the other hand, $R(\vec{\pi})$ is smallest if we transmit with the lowest cost for each data unit. Thus, a lower bound for $R(\vec{\pi})$ is

$$R_{\min}(\vec{\pi}') = R(\pi'_1, \ldots, \pi'_{\text{len}(\vec{\pi}')}, \underbrace{\mathbf{0}, \ldots, \mathbf{0}}_{L - \text{len}(\vec{\pi}')}) \qquad (15)$$

since $\mathbf{0}$ is the policy with the lowest cost. Hence, a lower bound for $J_\lambda(\vec{\pi})$ is

$$J_{\lambda,\min}(\vec{\pi}') = D_{\min}(\vec{\pi}') + \lambda R_{\min}(\vec{\pi}'). \qquad (16)$$

Pseudo-codes for the above branch and bound algorithms are given in [10], Appendix 6, 7, 8, 9. We denote the branch and bound algorithm for computing an optimal policy vector for a given rate constraint by CBBG ([10], Appendix 8).

The same kind of complexity analysis as in the single data unit case can be done, with the difference that the number of branches at a node is not equal to two anymore, but to $|\Pi|$ (resp. $|\Gamma|$) for the optimal policy vectors (resp. convex hull policy vectors). The worst-case complexity of the branch and bound algorithms for policy vectors is then $O(N2^N + L^2|\Pi|^L)$ (resp. $O(N2^N + L^2|\Gamma|^L)$).

V. Receiver-driven transmission

In the receiver-driven context [2], we have N time opportunities $r_0, \ldots, r_{N-1}$ at which the receiver can transmit a request packet to the sender. A *request policy* $\tau = (\tau(0), \ldots, \tau(N-1)) \in \{0,1\}^N$ for a data unit can be described as follows. If $\tau(i) = 0$, then the receiver should not send a request packet to the sender at opportunity r_i. If $\tau(i) = 1$, then the receiver should send a request packet to the sender at opportunity r_i if the data unit was not received, and it should not send a request packet otherwise. We define an error $\epsilon'(\tau)$ for request policy τ as the probability that no data unit reaches the receiver before a delivery deadline r_{DTS}, that is

$$\epsilon'(\tau) = \prod_{i:\tau(i)=1} P\{\overline{RTT} > r_{DTS} - r_i\}. \qquad (17)$$

We now have $\overline{RTT}$ instead of $\overline{FTT}$ since we have to include the transmission of the request packet to the sender. We also define a cost $\rho'(\tau)$ for request policy τ as the expected number of data unit transmissions from the sender to the receiver, that is

$$\rho'(\tau) = \sum_{i:\tau(i)=1} \left(\prod_{j<i:\tau(j)=1} P\{\overline{RTT} > r_i - r_j\} \right) (1-\epsilon_B). \qquad (18)$$

The sum is the same as the cost in the case of sender-driven operation, but on each request, the sender sends only with a probability of $(1-\epsilon_B)$ because the request packet is lost with probability ϵ_B. Now all above branch and bound algorithms can be used by replacing transmission policies π with request policies τ, error $\epsilon(\pi)$ with $\epsilon'(\tau)$, and cost $\rho(\pi)$ with $\rho'(\tau)$.

VI. Experimental results

In this section, we provide experimental results to illustrate the improvements allowed by our branch and bound algorithms. As in [2], the probability density function of FTT was modeled as a shifted Gamma distribution with rightward shift κ_F and parameters n_F and α_F [2], that is, for $t \geq \kappa_F$,

$$p_F(t) = \frac{\alpha_F}{\Gamma(n_F)} (\alpha_F(t - \kappa_F))^{n_F - 1} e^{-\alpha_F(t-\kappa_F)},$$

where Γ is the gamma function. Similarly, the probability density function of BTT was modeled as the shifted Gamma distribution

$$p_B(t) = \frac{\alpha_B}{\Gamma(n_B)} (\alpha_B(t - \kappa_B))^{n_B - 1} e^{-\alpha_B(t-\kappa_B)},$$

where κ_B is the rightward shift, and n_B and α_B are parameters.

A. Single data unit

We compare the branch and bound algorithm LBB of Section III-B.1 to the dynamic programming approach of [2] for computing a policy that minimizes the Lagrangian (3) for a fixed λ. We use the number of visited nodes in the decision process tree as the performance measure. This is a reasonable choice since both algorithms require a similar number of operations per node, with the difference that in the branch and bound algorithm, one more comparison is made to decide if the current branch should be pruned.

In practice, however, we seek the best policy for a given maximum cost, not for a given Lagrange multiplier. With the approach of [2], one must then combine the dynamic programming algorithm with the bisection method to find a convex hull solution for this given cost. A straightforward way to speed up this approach is to replace dynamic programming with our branch and bound technique. However, even with this improvement, this approach has two drawbacks. First, several iterations might be needed until a solution is found. Second, one cannot guarantee that the solution is optimal for the given cost since only convex hull solutions can be found. A better and faster approach is to use the branch and bound algorithm CBB of Section III-B.2. Figure 3 compares the complexity of respectively the dynamic programming approach of [2], algorithm LBB, and algorithm CBB. In the latter algorithm, the cost constraint was taken as the one corresponding to the solution of the first two algorithms. We provide results for two different Lagrange multipliers. The channel parameters were chosen as in [2]. Figure 4 shows the results for another set of channel parameters.

While dynamic programming must visit all $2^{N+1}-1$ nodes, the number of visited nodes is much smaller for the two branch and bound algorithms. Note also how algorithm CBB was faster than algorithm LBB.

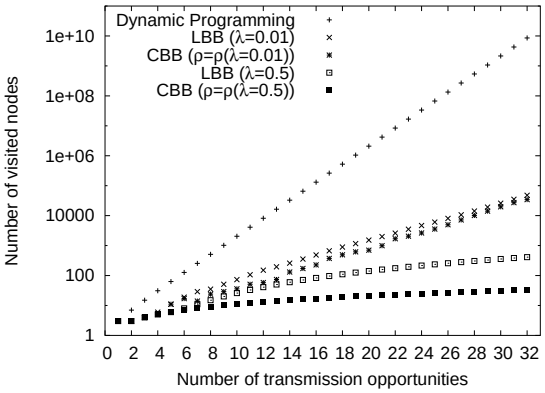

Fig. 3. Number of visited nodes as a function of the number of transmission opportunities for the dynamic programming algorithm and the branch and bound algorithms LBB and CBB. Results are shown for two Lagrange multipliers ($\lambda =$ 0.01 and 0.5) and channel parameters $\epsilon_F = \epsilon_B = 0.2, \kappa_F = \kappa_B = 25$ ms $, n_F = n_B = 2, 1/\alpha_F = 1/\alpha_B = 12.5$. The time interval between two opportunities is 50 ms.

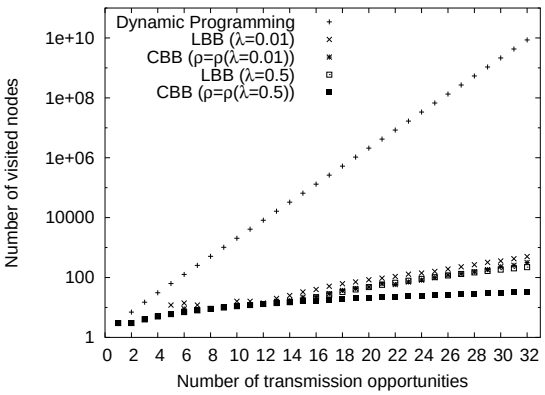

Fig. 4. Number of visited nodes as a function of the number of transmission opportunities for the dynamic programming algorithm and the branch and bound algorithms LBB and CBB. Results are shown for two Lagrange multipliers ($\lambda =$ 0.01 and 0.5) and channel parameters $\epsilon_F = \epsilon_B = 0.01, \kappa_F = \kappa_B = 25$ ms $, n_F = n_B = 8, 1/\alpha_F = 1/\alpha_B = 12.5$. The time interval between two opportunities is 50 ms.

B. Group of data units

We now compare the quality of the solutions found by the SA algorithm to the ones computed with the branch and bound algorithm for a set of data units and a given rate constraint (Algorithm CBBG in Section IV). We provide examples, two for MPEG1 and two for H.264, that show that the SA algorithm can perform poorly. As in [2], the stopping criterion for the SA algorithm is $J_\lambda(\vec{\pi}^{(k-1)}) = J_\lambda(\vec{\pi}^{(k)})$, and the single policies are changed in round-robin style ($l_k = ((k-1) \bmod L)+1$). The distortion values D_0 and ΔD_i, $i = 1, \ldots, L$, were computed as follows. Let X_i, $i = 1, \ldots, L$ denote the block in the luminance component of the original video sequence corresponding to packet i, Y_i, $i = 1, \ldots, L$ denote the block in the luminance component of the decoded sequence corresponding to packet i, G denote a block of constant grey value 128. Then $D_0 = \frac{1}{L}\sum_{i=1}^{L} \mathrm{MSE}(X_i, G)$ and $\Delta D_i = \frac{1}{L}(\mathrm{MSE}(X_i, G) - \mathrm{MSE}(X_i, Y_i))$, where MSE is the mean-square-error.

We first encoded the Foreman video sequence in CIF size according to the Video-CD standard (MPEG1, 1150 kilobits per second with a frame rate of 25 frames per second) and computed policy vectors for a group of ten frames (frame 13 to frame 22). The frame sequence was IBBPBBPBBP, where I-frames are independent, P-frames depend on the last P- or I-frame, and B-frames depend

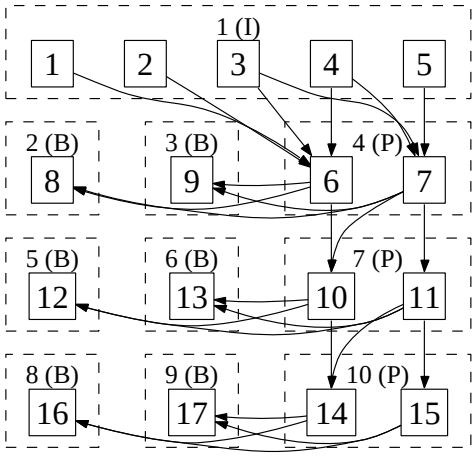

Fig. 5. Dependency graph of an H.264 group of frames. For simplicity, only the transitive reduction of the graph is shown. The boxes bounded by a dashed line indicate the frames with the frame numbers in playback order and the frame types in parentheses. The boxes bounded by a full line indicate the packets with the packet number in transmission order.

on the last and next P- or I-frame. The frame sizes (in bits) were $(B_1, \ldots, B_{10})$=(211048, 30252, 24996, 178508, 20820, 20292, 81988, 26820, 15164, 77676) and we assumed that each frame was transmitted in one packet. The distortions were $(D_0, \Delta D_1, \ldots, \Delta D_{10})$ =(5658.78, 560.59, 560.32, 562.07, 566.23, 567.47, 567.13, 567.41, 566.15, 566.30, 565.98). Finally, the number of transmission opportunities was $N = 8$, and the channel parameters were as in Fig. 3. For simplicity, we specify policies as sets of *send* actions. Thus, a policy π is now seen as a subset of $\{1, \ldots, N\}$ where i is in the subset if $\pi(i) = 1$.

The SA solution for Lagrange multiplier $\lambda = 0.0115$ was ({1}, $\emptyset$, {1}, {1,6}, {1}, {1}, {1,6}, {1}, {1,6}, {1}), yielding an expected rate of 756,566 bits and an expected MSE of 2421.35. The branch and bound algorithm was able to find a much better solution. Indeed, for rate constraint 756,566 bits, it found the solution ({1,5}, {1}, {1}, {1,5}, {1,5}, {1,4,7}, {1,5}, $\emptyset$, $\emptyset$, $\emptyset$), whose expected rate and MSE are 756,560 bits and 2289.82, respectively.

For $\lambda = 0.012$ and the same channel parameters, the SA solution was ($\emptyset$, $\emptyset$, $\emptyset$, {1}, {1}, {1}, {1}, {1}, $\emptyset$, {1}, $\emptyset$), yielding an expected rate of 341,768 bits and an expected MSE of 5658.78. For this rate constraint, the branch and bound algorithm found solution ({1,4,6}, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$), whose expected rate and MSE were 341,181 bits and 5102.68, respectively.

Note that for $\lambda = 0.012$, the SA solution transmits B- and P- frames without sending the I-frame they depend on, which is not a reasonable strategy.

We encoded the same group of ten frames with the H.264 reference encoder [15], version 9.3, at 384 kilobits per second with a frame rate of 30 frames per second using the main profile of H.264. Each frame was partitioned into slices such that each slice fits in a single packet with a maximum size of 1500 bytes. The encoder generated 17 packets whose dependency graph is shown in Fig. 5. The packet sizes (in bytes) were $(B_1, \ldots, B_{17})$=(1382, 1398, 1390, 1396, 1023, 1402, 1271, 774, 547, 1394, 664, 344, 341, 1402, 600, 232, 302). The distortions were $(D_0, \Delta D_1, \ldots, \Delta D_{17})$=(4042.92, 97.35, 133.11, 62.06, 71.78, 34.91, 297.09, 106.05, 399.10, 400.27, 330.35, 73.56, 404.24, 404.03, 356.14, 46.99, 403.06, 403.28). The number of transmission opportunities was $N = 5$, the channel parameters were as in Fig. 3.

The SA solution for Lagrange multiplier $\lambda = 0.44$ was ({1,4}, {1,4}, {1,4}, {1,4}, {1,4}, {1,4}, {1,4}, $\emptyset$, $\emptyset$, {1,4}, {1,4}, {1}, {1}, {1}, {1}, {1}, {1}), yielding an expected rate of 18923 bytes and an expected MSE of 2103.76. For the rate constraint of 18923 bytes, the branch and bound algorithm found the solution ({1,4}, {1,4}, {1,4}, {1,4}, {1,4,5}, {1,4}, {1,4}, {1,4}, {1,4}, {1,4}, {1,4}, {1,4}, {1,4}, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$), whose expected rate and MSE were 18854 bytes and 1923.71, respectively.

For $\lambda = 0.49$ and the same channel parameters, the SA solution was ({1}, {1}, {1}, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, {1,4}, {1,4}, {1}, {1}, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$), yielding an expected rate of 7710 bytes and an expected MSE of 3808.91. For the rate constraint of 7710 bytes, the branch and bound algorithm found the solution ({1}, {1,4}, {1}, {1}, $\emptyset$, {1}, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$) with an expected rate of 7509 bytes and an expected MSE of 3614.07.

Note how for $\lambda = 0.49$ the SA solution transmits packets without transmitting packets they depend on (e.g., packet 10 depends on packet 6 which is not sent).

VII. Conclusion

This work has three important contributions. First, it showed that the problem of rate-distortion optimized streaming of packetized multimedia in the sender-driven transmission over a single QoS network using retransmissions with feedback is NP-hard. This justifies the use of fast heuristic techniques such as the SA algorithm. Second, it gives a branch and bound algorithm for minimizing the Lagrangian for a single data unit. Our algorithm is much faster than the dynamic programming approach of [2]. Thus, it can be used to speed up the SA algorithm. Finally, the work introduces branch and bound algorithms for computing optimal policies for a group of interdependent data units. These algorithms are too time-consuming for online applications, but they can be used as an alternative to the SA algorithm in offline applications or to evaluate the quality of suboptimal solutions.

Acknowledgments

We thank Phil Chou and Jacob Chakareski for helpful discussions.

References

[1] D. Wu, Y.T. Hou, W. Zhu, Y.-Q. Zhang, and J.M. Peha, "Streaming video over the Internet: Approaches and directions", *IEEE Trans. Circuits and Sytems for Video Technology,* vol. 11, pp. 282–300, March 2001.

[2] P.A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media", Microsoft Research Technical Report MSR-TR-2001-35, Feb. 2001.

[3] T. Wiegand,G. Sullivan, and A. Luthra (Editors), "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 — ISO/IEC 14496-10 AVC)", JVT-G050r1, Geneva, Switzerland, May 2003.

[4] J. Chakareski, P. A. Chou, and B. Aazhang, "Computing rate-distortion optimized policies for streaming media to wireless clients", *Proc. DCC'02*, pp. 53–62, Snowbird, UT, April 2002.

[5] M. Kalman, P. Ramanathan, and B. Girod, "Rate-distortion optimized video streaming with multiple deadlines", *Proc. IEEE ICIP-03*, vol. 3, pp. 277–280, Barcelona, Sept. 2003.

[6] J. Chakareski and B. Girod, "Server diversity in rate-distortion optimized media streaming", *Proc. IEEE ICIP-03*, vol. 3, pp. 645–648, Barcelona, Sept. 2003.

[7] J. Chakareski and B. Girod, "Computing rate-distortion optimized policies for streaming media with rich acknowledgements", *Proc. DCC'04*, pp. 202–211, Snowbird, UT, April 2004.

[8] J. Chakareski, J. Apostolopoulos, and B. Girod, "Low-complexity rate-distortion optimized streaming", *Proc. IEEE ICIP-2004*, Singapore, Oct. 2004.

[9] A. Sehgal and P.A. Chou, "Cost distortion optimized streaming media over diffserv networks", *Proc. IEEE ICME-02*, vol. 1, pp. 857–860, Lausanne, Sept. 2002.

[10] M. Röder, J. Cardinal, and R. Hamzaoui, "On the complexity of rate-distortion optimal streaming of packetized media", Konstanzer Schriften in Mathematik und Informatik, No. 195, Jan. 2004, `http://www.inf.uni-konstanz.de/Preprints/`.

[11] F.P. Preparata and M.I. Shamos, *Computational Geometry*, Springer-Verlag, 1985.

[12] M.R. Garey and D.S. Johnson, *Computers and Intractability,* W.H. Freeman and Company, New York, 1979.

[13] D. Pisinger, *Algorithms for Knapsack Problems*, PhD thesis, University of Copenhagen, 1995.

[14] R. Szekli, *Stochastic Ordering and Dependence in Applied Probability*, Lecture Notes in Statistics 97, Springer-Verlag, 1995.

[15] H.264/AVC reference implementation, `http://bs.hhi.de/~suehring/tml/`.

Martin Röder received the Diploma degree in computer science from the University of Leipzig, Germany, in 2002. He is currently a Research Assistant at the Department of Computer and Information Science of the University of Konstanz, Germany. His research interests include convolutional codes, joint source-channel coding and error robust multimedia streaming.

Jean Cardinal was born in Brussels, Belgium, in 1976. He received the M.S. and Ph.D. degrees in computer science at the Université Libre de Bruxelles (ULB), Brussels, in 1998 and 2001, respectively. Between 1999 and 2001, he was a Research Fellow of the Belgian National Fund for Scientific Research (FNRS). Since 2002, he has been an Assistant Professor in the Computer Science Department of the Faculty of Sciences at ULB. His research interests include data compression, information theory and combinatorial algorithms.

Raouf Hamzaoui received the Maîtrise de mathématiques from the Faculty of Sciences of Tunis, Tunisia, in 1986, the M.Sc. degree in mathematics from the University of Montreal, Canada, in 1993, the Dr. rer. nat. degree from the Faculty of Applied Sciences of the University of Freiburg, Germany, in 1997, and the Habilitation degree in computer science from the University of Konstanz, Germany, in 2004. From 1998 to 2002, he was a Research Assistant with the Computer Science Department of the University of Leipzig, Germany. He is currently a Research Assistant with the Department of Computer and Information Science of the University of Konstanz. His research interests include data compression, error control systems, and algorithms.