On the complexity of rate-distortion optimal streaming of packetized media

Martin Röder[1], Jean Cardinal[2], Raouf Hamzaoui[1]

Abstract

We consider the problem of rate-distortion optimal streaming of packetized media with sender-driven transmission over a single-QoS 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 in practice much faster than the best known solution. 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 slower than the current state of the art, the heuristic sensitivity adaptation algorithm, but provide a significantly better rate-distortion performance in many real-world situations.

1 Introduction and preliminaries

One of the most popular frameworks for streaming packetized multimedia data over a packet erasure channel using feedback and retransmissions was introduced by Chou and Miao [1]. This framework enables a precise mathematical formulation of the problem of how and when to transmit a group of interdependent data units before a delivery deadline in a rate-distortion optimal way. Chou and Miao [1] first provide 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 propose a heuristic iterative descent method called sensitivity adaptation (SA), which optimizes the transmission policy for one data unit at a time. Chou and Miao's framework was also successfully used to handle other transmission scenarios [2, 3, 4, 5]. In this paper, we focus on sender-driven transmission over a single-QoS network. We first show the NP-hardness of the problem for a single data unit (Section 2.1). In Section 2.2, 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 [1]. By using our branch and bound algorithm instead of dynamic programming, one can significantly speed up the

[1]Universität Konstanz, FB Informatik und Informationswissenschaft, Fach M 697, 78457 Konstanz, Germany, {roeder,hamzaoui}@inf.uni-konstanz.de.

[2]Université Libre de Bruxelles, CP 212, B-1050, Brussels, Belgium, jcardin@ulb.ac.be.

SA algorithm. 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 3, we show that for a policy of a group of data units to be optimal, the policy of each data unit must be optimal. Then we give branch and bound algorithms for computing optimal policies for a group of data units. Section 4 gives experimental results for a single and a group of data units. This paper is a shortened version of a technical report [6], which contains the detailed proofs and algorithm descriptions that could not be included here due to space limitations.

In the remaining of this section, we recall the terminology and notation needed to describe the sender-driven transmission over a single-QoS network context [1]. 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 the partial order relation $\prec$, where $l' \prec l$ means that data unit l can be decoded only if data unit l' is decoded. We also use the notation $l' \preceq l$ if $l' \prec l$ or $l' = l$.

We denote by D_0 the distortion if no data unit is decoded, $B_l > 0$ the size in bytes of data unit l, and ΔD_l the amount by which the distortion is decreased if data unit l is decoded compared to the distortion if only the ancestors of l (i.e., all l' with $l' \prec l$) are decoded. 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 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 [1]. We call $\overline{FTT}$ the extended variable. The same extension can be done for BTT, yielding an extended random variable, $\overline{BTT}$. Finally, $\overline{RTT}$ will denote the sum $\overline{FTT} + \overline{BTT}$. Given a number of transmission opportunities at times $s_0, \ldots, s_{N-1}$ and a delivery deadline s_{DTS}, a transmission policy $\pi = (\pi(0), \ldots, \pi(N-1)) \in \{0,1\}^N$ of length $\text{len}(\pi) = N$ is used to describe the transmission of a single data unit. Here $\pi(i) = 0$ means that the data unit should not be sent at opportunity i, whereas $\pi(i) = 1$ means that the data unit should be sent at opportunity i if no acknowledgment packet was received from the feedback channel before time s_i.

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,

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

We also define a cost $\rho(\pi)$ for policy π as the expected number of data unit transmis-

sions. One can show (see [6] for a complete proof) that

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

The first problem discussed in this paper is the computation of rate-distortion optimal policies for the transmission of a data unit. A policy π^* is said to be *optimal* if there exists no policy π such that $\epsilon(\pi) \leq \epsilon(\pi^*)$ and $\rho(\pi) < \rho(\pi^*)$. Chou and Miao [1] claim that computing optimal policies is infeasible and propose to determine instead only those optimal policies π for which the cost-error points $(\rho(\pi), \epsilon(\pi))$ are on the lower convex hull of the set of all achievable (ρ, ϵ) points, i.e., that minimize the Lagrangian $J_\lambda(\pi) = \epsilon(\pi) + \lambda\rho(\pi),\ \lambda > 0$. We call such policies *convex hull* policies. For a given λ, brute force minimization of the Lagrangian requires $O(N^2 2^N)$ time. Chou and Miao [1] reduce the complexity of the minimization problem to $O(N2^N)$ 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 $\pi_i,\ i \in \{1, \ldots, L\}$ is the transmission policy for the ith data unit of the group. The expected transmission cost (the expected rate) for $\vec{\pi}$ is

$$R(\vec{\pi}) = \sum_{l=1}^{L} B_l \rho(\pi_l). \tag{3}$$

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

$$D(\vec{\pi}) = D_0 - \sum_{l=1}^{L} \Delta D_l \prod_{l' \preceq l} (1 - \epsilon(\pi_{l'})). \tag{4}$$

We say that 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 [1] propose to compute only those policy vectors $\vec{\pi}$ minimizing $J_\lambda(\vec{\pi}) = D(\vec{\pi}) + \lambda R(\vec{\pi})$ for $\lambda > 0$. We call these policy vectors convex hull policy vectors. To minimize J_λ for a given λ, Chou and Miao [1] apply the SA algorithm, a heuristic iterative descent method. 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 $\vec{\pi}^{(0)} = (\pi_1^{(0)}, \ldots, \pi_L^{(0)})$, a sequence of policy vectors $\vec{\pi}^{(k)}$ is computed 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

$$\begin{aligned} \pi_i^{(k)} &= \arg\min_{\pi_l} J_\lambda(\pi_1^{(k)}, \ldots, \pi_{l-1}^{(k)}, \pi_l, \pi_{l+1}^{(k)}, \ldots, \pi_L^{(k)}) & (5) \\ &= \arg\min_{\pi_l} S_l^{(k)} \epsilon(\pi_l) + \lambda B_l \rho(\pi_l), & (6) \end{aligned}$$

where $S_l^{(k)} = \sum_{l \preceq l'} \Delta D_{l'} \prod_{l'' \prec l'} (1 - \epsilon(\pi_{l''}^{(k)}))$. Since the sequence $J_\lambda(\vec{\pi}^{(k)})$ is nonincreasing and bounded below by zero, it converges to a local minimum. The complexity of the SA algorithm is $O(MN2^N)$, where M is the number of iterations of the algorithm.

2 Single data unit

2.1 NP-hardness

We show that finding an optimal transmission policy for a single data unit is NP-hard by using a reduction from the knapsack problem [7]. We first formally define the considered problem.

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

For simplicity, and without loss of generality, we assumed in Definition 1 that the optimality constraint is slightly less restrictive than in Section 1. We also assumed that 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, that is, the finite set of probabilities $P\{FTT > s_{DTS} - s_i\}$ and $P\{RTT > s_i - s_j\}$ is the only information we actually need about FTT and RTT.

Lemma 1. *Let* π^* *be an optimal solution for a given instance of problem* POLICY *with* $\pi^* \neq (0, \ldots, 0)$. *Then there exists an optimal solution* π *for the same instance with* $\pi(0) = 1$.

Proof. By contradiction. See [6] for details. □

Definition 2 (KNAPSACK). *An instance of problem* KNAPSACK *consists of a positive integer* m, *positive real values* w_i *and* v_i *for each* $i \in \{1, \ldots, m\}$, *and a positive real number* B. *An optimal solution of* KNAPSACK *is a set* $S \subseteq \{1, \ldots, m\}$ *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 [7, 8].

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

$$\begin{aligned} P\{FTT \leq 1 - \frac{i}{N}\} &= 1 - 2^{-w_i},\ i \in \{1, \ldots, N-1\}, \\ P\{FTT \leq \delta + 1\} &= 1 - \mu, \\ P\{RTT \leq \delta\} &= 0, \qquad (7) \\ P\{RTT \leq \delta + \frac{i}{N}\} &= 1 - v_i,\ i \in \{1, \ldots, N-1\} \qquad (8) \end{aligned}$$

where $0 < \mu < \min\{2^{-w_1}, v_{N-1}, 2^B\}$. 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. It is not difficult to check that the two cumulative distribution functions defined in this way are well-behaved nondecreasing functions taking values in $[0, 1]$. Also from the stochastic order relation ([9], Theorem A, p. 6), we can show that there exist two random variables FTT and RTT having the described cumulative distribution functions and such that $FTT \leq RTT$. The reader is referred to [6] for more details.

Finally, we set $\epsilon^* = \mu/2^B$, which belongs to $[0, 1]$ from the choice of μ.

Now we show that this instance of POLICY is equivalent to the original instance of KNAPSACK. Indeed, an optimal solution π of the first problem yields an optimal solution of the second by letting $S = \{i > 0 : \pi(i) = 1\}$, and conversely, subset S yields an optimal solution π for POLICY.

Let us first detail (1) with respect to the assignments above:

$$\epsilon(\pi) = \prod_{i:\pi(i)=1} P\{FTT > s_{DTS} - s_i\} \leq \epsilon^* \Leftrightarrow$$

$$(1 - P\{FTT \leq \delta + 1\}) \prod_{i>0:\pi(i)=1} \left(1 - P\{FTT \leq 1 - \frac{i}{N}\}\right) \leq \frac{\mu}{2^B} \Leftrightarrow \quad (9)$$

$$\mu \prod_{i>0:\pi(i)=1} 2^{-w_i} \leq \frac{\mu}{2^B} \Leftrightarrow \sum_{i>0:\pi(i)=1} w_i \geq B. \quad (10)$$

Here we used Lemma 1 in (9) to include the probability $P\{FTT > s_{DTS} - s_0\}$ in the product. From (7)-(8) we have $P\{RTT > s_i - s_j\} = \begin{cases} v_i \text{ if } j = 0 \\ 1 \text{ otherwise} \end{cases}$.
Now if we expand (2) with respect to the previous assignments and apply Lemma 1 again, we obtain

$$\rho(\pi) = \sum_{i:\pi(i)=1} \left(\prod_{j<i:\pi(j)=1} P\{RTT > s_i - s_j\} \right) = 1 + \sum_{i>0:\pi(i)=1} v_i. \quad (11)$$

Inequality (10) and equation (11) are the same as in the KNAPSACK problem except for the constant additional term in (11), which does not modify the optimal solution. Hence POLICY is at least as hard as KNAPSACK. □

2.2 Branch and bound

In this section, we propose a class of branch and bound algorithms to compute convex hull and optimal policies.

2.2.1 Convex hull policies

The algorithm exploits the following 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

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

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

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

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

Using the above bound, we first propose a branch and bound algorithm to compute a convex hull policy for a fixed Lagrange multiplier λ. The policy prefixes of size n are 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 prefixes 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. 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. Let π'_0 be π' with 0 appended, and π'_1 be π' with 1 appended. We then compute $J_{\lambda,\min}(\pi'_k)$ 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 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. The optimal policy will be in π when the algorithm ends.

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. 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 belongs 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 $\Gamma(i)$; so we assume that the ordering satisfies $\rho(\Gamma(i)) \leq \rho(\Gamma(i+1))$, $i = 0, \ldots, |\Gamma| - 2$ 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' belongs to the lower convex hull Γ. The test consists of first finding

a k such that $\rho(\Gamma(k)) < \rho_{\min}(\pi') \leq \rho(\Gamma(k+1))$. If $(\rho_{\min}(\pi'), \epsilon_{\min}(\pi'))$ is above the line joining $\Gamma(k)$ and $\Gamma(k+1)$, the function equals "yes". Otherwise, or if no suitable k can be found, it equals "no". 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 the left and right neighbors of the inserted point iteratively (see [6]).

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 (the current lower convex hull) with $\Gamma = \{(0, \ldots, 0), (1, \ldots, 1)\}$. We then start the recursive part of the algorithm with prefix $\pi' = (1)$. In each single recursion, we determine $\pi'_k,\ k = 0, 1$ 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 lower convex hull of the policies with length N.

2.2.2 Optimal policies

The branch and bound approach to compute the 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 a policy 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.

To find all optimal points, we use the same algorithm as for the lower convex hull. We only need to change test function *CHTest* and insertion procedure *CHInsert*. We now use a test function *OPTest(*Π*,*π'*)* that checks if policies with prefix π' are optimal among the set Π of all current optimal policies, and a function *OPInsert(*Π*,*π*)*.

The worst-case complexity for the computation of Γ (resp. Π) is $O(N2^N)$ [6]. Again, the pruning procedure *CHTest* (resp. *OPTest*) should yield substantial acceleration ratios in practice.

3 Group of data units

We study in this section the complexity of optimal rate-distortion streaming for a group of interdependent data units. We first provide two important results that give relationships between optimal (resp. convex hull) policy vectors and optimal (resp. convex hull) policies.

Lemma 2. *Let* $\vec{\pi}^* = (\pi_1^*, \ldots, \pi_L^*)$ *be an optimal policy vector. Then all policies* $\pi_1^*, \ldots, \pi_L^*$ *are optimal.*

Lemma 3. *Let* $\vec{\pi}^* = (\pi_1^*, \ldots, \pi_L^*)$ *be a convex hull policy vector. Then all policies* $\pi_1^*, \ldots, \pi_L^*$ *are convex hull policies.*

Both lemmas can be proved by contradiction (see [6]). Note that 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.

Lemma 2 allows us to break the problem of finding an optimal policy vector for a group of data units into two parts. First, use the branch and bound algorithm of Section 2.2.2 to compute the set Π of all optimal 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. We can derive lower bounds for pruning the recursion tree as in the single policy case.

Let $\Pi(i)$ be the ith policy of Π in non-decreasing cost and non-increasing error order. Then $\Pi(0)$ is the policy with the highest error and lowest cost (i.e. $(0,\ldots,0)$) and $\Pi(|\Pi|-1)$ is the policy with the lowest error and highest cost (i.e. $(1,\ldots,1)$). As in the single data unit case, if we know that policy vector $\vec{\pi}$ begins with prefix $\vec{\pi}' = (\pi'_1, \pi'_2, \ldots, \pi'_{\text{len}(\vec{\pi}')})$, $\pi'_i = \pi_i$ for $i = 1,\ldots,\text{len}(\vec{\pi}')$, $\text{len}(\vec{\pi}') \leq L$, we can give bounds for $D(\vec{\pi})$ and $R(\vec{\pi})$. Since $D(\vec{\pi})$ is smallest if we transmit with the smallest error for each data unit, a lower bound for $D(\vec{\pi})$ is

$$D_{\min}(\vec{\pi}') = D(\pi'_1, \ldots, \pi'_{\text{len}(\vec{\pi}')}, \underbrace{\Pi(|\Pi|-1), \ldots, \Pi(|\Pi|-1)}_{L-\text{len}(\vec{\pi}') \text{ times}}). \tag{12}$$

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

$$R_{\min}(\vec{\pi}') = R(\pi'_1, \ldots, \pi'_{\text{len}(\vec{\pi}')}, \underbrace{\Pi(0), \ldots, \Pi(0)}_{L-\text{len}(\vec{\pi}') \text{ times}}). \tag{13}$$

A lower bound for the expected Lagrangian $J_\lambda(\vec{\pi})$ is $J_{\lambda,\min}(\vec{\pi}') = D_{\min}(\vec{\pi}') + \lambda R_{\min}(\vec{\pi}')$.

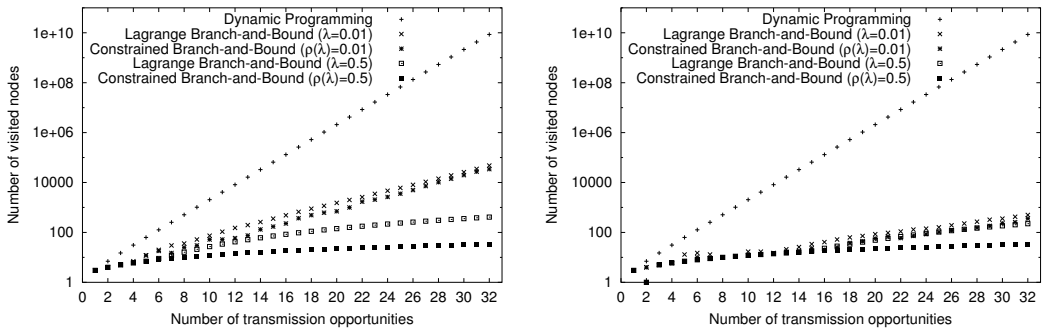
Exploiting Lemmas 2 and 3, and bounds (12), (13), and $J_{\lambda,\min}$, we can use branch and bound methods as in the single data unit case.

The same kind of complexity analysis as in the single data unit case can be done, with the difference that the branching factor 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 computation of an optimal policy vector, ignoring all pruning steps, is then $O(N2^N + L^2|\Pi|^L)$ (resp. $O(N2^N + L^2|\Gamma|^L)$).

4 Experimental results

In this section, we provide experimental results to illustrate the improvements allowed by our branch and bound algorithms. As in [1], 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, that is, for $t \geq \kappa_F$, we have $p_F(t) = \frac{\alpha_F}{\Gamma(n_F)}(\alpha_F(t - \kappa_F))^{n_F-1}e^{-\alpha_F(t-\kappa_F)}$, where Γ is the gamma function. Similarly, the probability density function of BTT was modeled as the shifted Gamma distribution with parameters κ_B, n_B, and α_B.

We first compare the branch and bound algorithm to the dynamic programming approach of [1] for computing a policy that minimizes the Lagrangian J_λ for a fixed



(a) (b)

Figure 1: Number of visited nodes as a function of the number of transmission opportunities for the dynamic programming algorithm and the branch and bound algorithms of Section 2.2.1 and 2.2.2. The results are shown for Lagrange multiplier $\lambda = 0.01$ and 0.5. The number of transmission opportunities is $N = 8$, the time interval between two opportunities is 50 ms. The channel conditions are (a) $\epsilon_F = \epsilon_B = 0.2, \kappa_F = \kappa_B = 25$ ms $, n_F = n_B = 2, 1/\alpha_F = 1/\alpha_B = 12.5$ and (b) $\epsilon_F = \epsilon_B = 0.01, \kappa_F = \kappa_B = 25$ ms $, n_F = n_B = 8, 1/\alpha_F = 1/\alpha_B = 12.5$.

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 [1], one must then combine the dynamic programming algorithm with the bisection method to find a convex hull solution for this given cost. However, this approach might require many iterations and does not always yield optimal solutions. A better and faster approach is to use the branch and bound algorithm of Section 2.2.2 that finds an optimal transmission policy for a given cost. Fig. 1(a) compares the complexity of respectively the dynamic programming approach of [1], the branch and bound algorithm minimizing the Lagrangian cost J_λ of Section 2.2.1 (curves denoted by "Lagrange branch and bound"), and the algorithm of Section 2.2.2 (curves denoted by "Constrained branch and bound"). 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 [1]. Fig. 1(b) shows the results for another set of channel parameters. While dynamic programming has an $O(N2^N)$ time complexity, the results show a much lower complexity for the two branch and bound algorithms. Note also how the branch and bound algorithm of Section 2.2.2 had lower complexity than the one of Section 2.2.1.

The branch and bound algorithms for computing optimal policy vectors are too time-consuming for online applications. But they can be used in offline applications or to evaluate the quality of heuristic solutions. We compared 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. Table 1 shows that the gaps may be significant between SA and optimal solutions

SA			branch and bound	
λ	rate (bits)	Y-PSNR (dB)	rate (bits)	Y-PSNR (dB)
$6.4 \cdot 10^{-5}$	756,566	29.97	756,560	30.67
$7.2 \cdot 10^{-5}$	341,768	11.78	341,187	15.10

Table 1: SA and optimal solutions found by branch and bound for 10 frames of an MPEG1 encoded Foreman video sequence with dependencies IBBPBBPBBP. Here $(B_1, \ldots, B_{10})$=(211048, 30252, 24996, 178508, 20820, 20292, 81988, 26820, 15164, 77676) in bits, $(D_0, \Delta D_1, \ldots, \Delta D_{10})$ =(11.78, 3.35, 3.01, 3.06, 3.53, 2.94, 2.93, 3.26, 2.98, 3.08, 3.24) in dB. The number of transmission opportunities is $N = 8$, and the channel parameters are as in Fig. 1(a).

5 Conclusion

This paper has three important contributions. First, it shows that the problem of optimal rate-distortion 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. The algorithm is much faster than the dynamic programming approach of [1]. Thus, it can be used to speed up the SA algorithm. Finally, the paper introduces branch and bound algorithms for computing optimal policies for a group of interdependent data units. These algorithms can be used in offline applications or to evaluate the quality of suboptimal solutions.

Acknowledgment. We thank Phil Chou for helpful discussions.

References

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

[2] J. Chakareski, P. A. Chou, and B. Aazhang, *Computing rate-distortion optimized policies for streaming media to wireless clients*, *Proc. DCC'02*, Utah, April 2002.

[3] J. Chakareski and B. Girod, *Server diversity in rate-distortion optimized media streaming*, *Proc. IEEE ICIP'03*, Barcelona, Sept. 2003.

[4] M. Kalman, P. Ramanathan, and B. Girod, *Rate-Distortion optimized video streaming with multiple deadlines*, *Proc. IEEE ICIP'03*, Barcelona, Sept. 2003.

[5] G. Cheung and C. Chan, *Jointly optimal reference frame & quality of service selection for H.26L video coding over lossy networks*, *Proc. IEEE ICME'03*, Baltimore, July 2003.

[6] 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/`.

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

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

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