Computing the Capacity of Rewritable Memories

Christoph Bunte and Amos Lapidoth
Signal and Information Processing Laboratory
ETH Zurich
Email: {bunte, lapidoth}@isi.ee.ethz.ch

***Abstract*—We propose an algorithm for computing the capacity of discrete rewritable storage devices subject to a constraint on the maximal number of rewrite operations. The linchpin is that—although the number of writing strategies is exponential in the maximal number of allowed rewrites—linear functionals of the probabilities they induce on the output space can be efficiently maximized using Dynamic Programming.**

I. Introduction

We address the numerical computation of the capacity of rewritable memory cells subject to a constraint on the maximal number of allowed rewrites. For a general information theoretic model for rewritable memories see the recent work by Lastras-Montaño et al. [1]. And for a study of the capacity of rewritable memory cells with continuous alphabets see [1]–[6]. For some related results on discrete write channels see [7].

A (noisy) memory cell is modeled as a discrete memoryless channel (DMC) with input alphabet $\mathcal{X} = \{1, \ldots, |\mathcal{X}|\}$, output alphabet $\mathcal{Y} = \{1, \ldots, |\mathcal{Y}|\}$, transition law $W(y|x)$, and a maximal number of rewrites L, where L is a nonnegative integer. The process of using a rewritable memory cell is specified by a sequence of inputs $x_0, \ldots, x_{\mathsf{L}}$ from $\mathcal{X}$ and a sequence of target sets $\mathcal{T}_0, \ldots, \mathcal{T}_{\mathsf{L}} \subseteq \mathcal{Y}$ with $\mathcal{T}_{\mathsf{L}} = \mathcal{Y}$. In the initial write iteration the cell is fed x_0 and produces Y_0 according to the law $W(\cdot|x_0)$. If Y_0 is in $\mathcal{T}_0$, the storer halts. Otherwise it feeds x_1. In general, the storer halts after feeding the k-th input symbol x_k if, and only if, Y_k is in $\mathcal{T}_k$, where Y_k is the output of the channel induced by x_k. Otherwise, it feeds x_{k+1}. Here $k = 0, \ldots, \mathsf{L}$ because the condition $\mathcal{T}_{\mathsf{L}} = \mathcal{Y}$ guarantees that the storer will halt after at most L rewrite iterations.

After the storage process is complete, the reader observes Y_N, where $N = \min\{k \geq 0 : Y_k \in \mathcal{T}_k\}$; it does not observe $Y_0, \ldots, Y_{N-1}$. We refer to the pair $x_0, \ldots, x_{\mathsf{L}}$ and $\mathcal{T}_0, \ldots, \mathcal{T}_{\mathsf{L}}$ as a "strategy."

More generally, we could allow for randomization in the storage process, but we shall not because this does not increase the storage capacity. Also, one could allow $(x_k, \mathcal{T}_k)$ to depend on $Y_0, \ldots, Y_{k-1}$, but this too does not increase the storage capacity. We thus only consider strategies that are characterized by an $(\mathsf{L}+1)$-tuple

$$s = \big((x_0, \mathcal{T}_0), \ldots, (x_{\mathsf{L}}, \mathcal{T}_{\mathsf{L}})\big),$$

with $\mathcal{T}_{\mathsf{L}} = \mathcal{Y}$. We denote the set of all such strategies by $\mathcal{S}$. Every $s \in \mathcal{S}$ induces a distribution of Y_N. This specifies a memoryless channel with input alphabet $\mathcal{S}$, output alphabet $\mathcal{Y}$, and capacity

$$C_{\mathsf{L}} = \max_{S} I(S; Y_N), \quad \mathsf{L} = 0, 1, \ldots,$$

where the maximization is over the space of distributions on $\mathcal{S}$.

Computing C_{L} for increasing values of L is challenging because the cardinality of $\mathcal{S}$ grows exponentially with L. This renders the computation of C_{L} using algorithms that optimize over the space of input distributions (e.g., the Blahut-Arimoto algorithm [8], [9]) intractable.[1] The algorithm we propose works primarily on the space of output distributions, which is relatively small and does not depend on L.

For every strategy $s \in \mathcal{S}$ let $\mathbf{w}(s)$ denote the induced distribution of Y_N. Taking the convex hull of the set

$$\mathcal{W} \triangleq \{\mathbf{w}(s) : s \in \mathcal{S}\}, \tag{1}$$

of all such distributions yields a convex polytope

$$\mathcal{P} \triangleq \operatorname{conv}(\mathcal{W}). \tag{2}$$

To achieve capacity, only strategies corresponding to the extreme points (vertices) of $\mathcal{P}$ are required. This can be seen from the Kuhn-Tucker conditions [11, Theorem 4.5.1] and the convexity of relative entropy. Since the vertices of $\mathcal{P}$ are in general difficult to determine, a promising approach is to approximate $\mathcal{P}$ by simpler convex polytopes whose vertices are readily computable. This is a common approach for concave minimization problems [12]. The key is to find suitable "cutting hyperplanes" to separate points from the feasible set. We show how to derive such hyperplanes using the Kuhn-Tucker conditions and properties of relative entropy.

The rest of this paper is organized as follows. In Section II we introduce our notation; in section III we show that one can restrict attention to strategies with certain properties, and we upper-bound the number of vertices of $\mathcal{P}$; in Section IV we develop the algorithm and prove its convergence. Finally, in Section V, we briefly discuss the implementation and performance of the algorithm.

II. Notation

We use uppercase letters for random variables and lowercase letters for their realizations. We use boldface letters for (deterministic) vectors. The i-th component of the vector $\mathbf{p}$ is denoted by $p^{(i)}$. The notation $\log \frac{\mathbf{p}}{\mathbf{q}}$ is to be understood componentwise:

$$\log \frac{\mathbf{p}}{\mathbf{q}} \triangleq \left(\log \frac{p^{(1)}}{q^{(1)}}, \ldots, \log \frac{p^{(n)}}{q^{(n)}}\right)^{\mathsf{T}}.$$

The standard inner product between $\mathbf{p}$ and $\mathbf{q}$ is denoted $\langle \mathbf{p}, \mathbf{q} \rangle$.

[1] This includes algorithms that are optimized for large input alphabets, e.g., [10], because the exponential growth of the cardinality of $\mathcal{S}$ in L makes it infeasible to compute all the entries of the transition matrix.

978-1-4577-0595-3/11/$26.00 ©2011 IEEE

III. On Capacity-Achieving Strategies and the Number of Extreme Points of $\mathcal{P}$

The following result is presented without proof.

Proposition 1. *Let $\mathcal{S}_0$ denote the subset of strategies that satisfy all of the following conditions.*

1) *Nondecreasing target sets: $\mathcal{T}_0 \subseteq \mathcal{T}_1 \subseteq \cdots \subseteq \mathcal{T}_{\mathsf{L}}$;*
2) *nontrivial target sets: $\mathcal{T}_0 \neq \emptyset$ and $\mathcal{T}_k \neq \mathcal{Y}$ for $k < \mathsf{L}$;*
3) *nondecreasing probability of stopping:*

$$\Pr(Y_{k+1} \in \mathcal{T}_{k+1}) \geq \Pr(Y_k \in \mathcal{T}_k);$$

4) *non-cycling pairs $(x_k, \mathcal{T}_k)$: $(x_{k+1}, \mathcal{T}_{k+1}) \neq (x_k, \mathcal{T}_k)$ implies $(x_{k'}, \mathcal{T}_{k'}) \neq (x_k, \mathcal{T}_k)$ for all $k' > k$.*

Then the vertices of $\mathcal{P}$ form a subset of $\{\mathbf{w}(s) : s \in \mathcal{S}_0\}$. In particular, it suffices to take strategies in $\mathcal{S}_0$ to achieve the capacity.

Corollary 1. *The number of vertices of $\mathcal{P}$ is upper-bounded by a polynomial in* L.

IV. An Algorithm for Computing C_{L}

The algorithm presented in this section exploits the following fact.

Proposition 2. *For every $\boldsymbol{\alpha} \in \mathbb{R}^{|\mathcal{Y}|}$ one can find a $\mathbf{w}^\star \in \mathcal{W}$ such that*

$$\langle \mathbf{w}^\star, \boldsymbol{\alpha} \rangle = \max_{\mathbf{w} \in \mathcal{W}} \langle \mathbf{w}, \boldsymbol{\alpha} \rangle ,$$

with linear complexity in L.

This can be proved by constructing a simple Dynamic Programming algorithm that carries out the maximization in linear time. The details are omitted.

To avoid some technical issues related to the discontinuity of relative entropy, we require that the channel law be positive:

$$W(y|x) > 0, \quad \text{for all } x \in \mathcal{X},\, y \in \mathcal{Y}. \tag{3}$$

This guarantees that $\mathcal{W}$ is included in the relative interior of the probability simplex in $\mathbb{R}^{|\mathcal{Y}|}$.

If $\mathsf{L} = 0$, then C_{L} is equal to the capacity of the channel $W(y|x)$. Thus, we shall assume throughout that $\mathsf{L} \geq 1$. In particular, it can be shown that this implies $C_{\mathsf{L}} > 0$ [7].

Consider the following dual expression for the capacity (see the comment on page 142 of [13]),

$$C_{\mathsf{L}} = \min_{\mathbf{r}} \max_{s \in \mathcal{S}} D(\mathbf{w}(s)||\mathbf{r}),$$

where the minimization is over the space of distributions on $\mathcal{Y}$. By the convexity of relative entropy, this may be written as

$$C_{\mathsf{L}} = \min_{\mathbf{r}} \max_{\mathbf{p} \in \mathcal{P}} D(\mathbf{p}||\mathbf{r}). \tag{4}$$

Considering (4), the idea is to generate a decreasing sequence of convex outer polytopes

$$\mathcal{P}_0 \supset \mathcal{P}_1 \supset \cdots \supset \mathcal{P}_k \supseteq \mathcal{P},$$

to obtain a nonincreasing sequence of upper bounds on C_{L}:

$$u_k \triangleq \min_{\mathbf{r}} \max_{\mathbf{p} \in \mathcal{P}_k} D(\mathbf{p}||\mathbf{r}), \quad k = 0, 1, \ldots \tag{5}$$

Let $\mathcal{V}_k$ denote the set of vertices of $\mathcal{P}_k$. By the convexity of the mapping $\mathbf{p} \mapsto D(\mathbf{p}||\mathbf{r})$,

$$u_k = \min_{\mathbf{r}} \max_{\mathbf{v} \in \mathcal{V}_k} D(\mathbf{v}||\mathbf{r}), \quad k = 0, 1, \ldots \tag{6}$$

Thus, u_k is the capacity of a DMC with $|\mathcal{V}_k|$ inputs whose transition matrix has as columns the elements of $\mathcal{V}_k$. In practice, u_k can be computed using, e.g., the Blahut-Arimoto or Cutting-Plane algorithm [14].

We now discuss the construction of the polytopes $\mathcal{P}_k$. To this end, we shall frequently use the following inequality.

Proposition 3. *If $\mathbf{p}, \mathbf{q}, \mathbf{r}$ are probability vectors in $\mathbb{R}^d$, then*

$$\left\langle \mathbf{p}, \log \frac{\mathbf{q}}{\mathbf{r}} \right\rangle \leq D(\mathbf{p}||\mathbf{r}),$$

with equality if, and only if, $\mathbf{p} = \mathbf{q}$.

Proof: This follows directly from the fact that $D(\mathbf{p}||\mathbf{q}) \geq 0$, with equality if, and only if, $\mathbf{p} = \mathbf{q}$. ∎

We initialize the algorithm with the outer polytope

$$\mathcal{P}_0 = \left\{ \mathbf{p} \in \mathbb{R}^{|\mathcal{Y}|} : p^{(i)} \geq \epsilon \; \forall i, \sum_{i=1}^{|\mathcal{Y}|} p^{(i)} = 1 \right\}, \tag{7}$$

where $\epsilon > 0$ is chosen sufficiently small so that $\mathcal{W} \subseteq \mathcal{P}_0$ (and hence $\mathcal{P} \subseteq \mathcal{P}_0$). This is possible by (3). Note that if $\mathbf{p}$ and $\mathbf{q}$ are in $\mathcal{P}_0$, then $\log \frac{\mathbf{p}}{\mathbf{q}}$ is well-defined. Moreover, relative entropy is continuous on $\mathcal{P}_0 \times \mathcal{P}_0$.

Suppose we have computed $\mathcal{P}_k$. Let $\mathbf{r}_k$ denote the unique capacity-achieving output distribution for $\mathcal{P}_k$. By [11, Corollary 3, p. 96], there exists a number

$$m_k \in \{2, \ldots, |\mathcal{Y}|\} \tag{8}$$

and a distribution $\mathbf{q}_k$ on m_k distinct elements of $\mathcal{V}_k$,

$$\mathbf{v}_{k,1}, \ldots \mathbf{v}_{k,m_k},$$

such that

$$\mathbf{r}_k = \sum_{j=1}^{m_k} q_k^{(j)} \mathbf{v}_{k,j}, \tag{9}$$

and such that

$$q_k^{(j)} > 0, \quad j = 1, \ldots, m_k.$$

Thus, $\mathbf{q}_k$ corresponds to a capacity-achieving input distribution for $\mathcal{P}_k$ with m_k mass points, and $\mathbf{v}_{k,1}, \ldots, \mathbf{v}_{k,m_k}$ are the output distributions induced by the corresponding m_k inputs. By the Kuhn-Tucker conditions,

$$D(\mathbf{v}_{k,j}||\mathbf{r}_k) = u_k, \quad j = 1, \ldots, m_k, \tag{10}$$

and

$$D(\mathbf{p}||\mathbf{r}_k) \leq u_k, \quad \mathbf{p} \in \mathcal{P}_k, \tag{11}$$

so

$$D(\mathbf{v}_{k,j}||\mathbf{r}_k) = \max_{\mathbf{p} \in \mathcal{P}_k} D(\mathbf{p}||\mathbf{r}_k), \quad j = 1, \ldots, m_k. \tag{12}$$

The following proposition shows that, once $\mathbf{r}_k$ is computed, finding a linear functional that is maximized uniquely over $\mathcal{P}_k$ by $\mathbf{v}_{k,j}$ is easy.

Proposition 4. *Let $j \in \{1, \ldots, m_k\}$. If $\mathbf{p} \in \mathcal{P}_k$, then*

$$\left\langle \mathbf{p}, \log \frac{\mathbf{v}_{k,j}}{\mathbf{r}_k} \right\rangle \leq D(\mathbf{v}_{k,j} || \mathbf{r}_k),$$

with equality if, and only if, $\mathbf{p} = \mathbf{v}_{k,j}$.

Proof: For every $\mathbf{p} \in \mathcal{P}_k$ and every $j \in \{1, \ldots, m_k\}$,

$$\left\langle \mathbf{p}, \log \frac{\mathbf{v}_{k,j}}{\mathbf{r}_k} \right\rangle \leq D(\mathbf{p} || \mathbf{r}_k) \leq u_k = D(\mathbf{v}_{k,j} || \mathbf{r}_k),$$

where we used Proposition 3 in the first inequality, (11) in the second inequality, and (10) in the equality. By Proposition 3, we have equality only if $\mathbf{p} = \mathbf{v}_{k,j}$. ∎

A corollary to Proposition 4 provides us with a simple test to check whether $\mathbf{v}_{k,j} \in \mathcal{P}$.

Corollary 2. *For every $j \in \{1, \ldots, m_k\}$,*

$$\max_{s \in \mathcal{S}} \left\langle \mathbf{w}(s), \log \frac{\mathbf{v}_{k,j}}{\mathbf{r}_k} \right\rangle \leq D(\mathbf{v}_{k,j} || \mathbf{r}_k), \tag{13}$$

with equality if, and only if, $\mathbf{v}_{k,j} \in \mathcal{P}$.

Proof: If $\mathbf{v}_{k,j} \notin \mathcal{P}$, then $\mathbf{w}(s) \neq \mathbf{v}_{k,j}$ for all $s \in \mathcal{S}$. Since $\mathbf{w}(s) \in \mathcal{P} \subseteq \mathcal{P}_k$, strict inequality in (13) follows from Proposition 4. If $\mathbf{v}_{k,j} \in \mathcal{P}$, then $\mathbf{v}_{k,j} = \mathbf{w}(s_0)$ for some $s_0 \in \mathcal{S}$ because $\mathbf{v}_{k,j}$ is an extreme point of $\mathcal{P}_k$. Thus,

$$\max_{s \in \mathcal{S}} \left\langle \mathbf{w}(s), \log \frac{\mathbf{v}_{k,j}}{\mathbf{r}_k} \right\rangle \geq \left\langle \mathbf{w}(s_0), \log \frac{\mathbf{v}_{k,j}}{\mathbf{r}_k} \right\rangle = D(\mathbf{v}_{k,j} || \mathbf{r}_k).$$

But the reverse inequality holds by Proposition 4. ∎

Note that Proposition 2 asserts that the maximization in (13) can be carried out efficiently. Another corollary to Proposition 4 is the following geometric observation.

Corollary 3. *If $\mathbf{v}_{k,j} \notin \mathcal{P}$, then the supporting hyperplane of $\mathcal{P}$ given by*

$$\left\{ \mathbf{p} : \left\langle \mathbf{p}, \log \frac{\mathbf{v}_{k,j}}{\mathbf{r}_k} \right\rangle = \max_{s \in \mathcal{S}} \left\langle \mathbf{w}(s), \log \frac{\mathbf{v}_{k,j}}{\mathbf{r}_k} \right\rangle \right\},$$

strictly separates $\mathbf{v}_{k,j}$ from $\mathcal{P}$.

Testing the vertices $\mathbf{v}_{k,1}, \ldots, \mathbf{v}_{k,m_k}$ for membership in $\mathcal{P}$ using Corollary 2, we obtain a subset of $\mathcal{V}_k$,

$$\mathcal{V}'_k \triangleq \big\{ \mathbf{v}_{k,j} : j \in \{1, \ldots, m_k\},\ \mathbf{v}_{k,j} \notin \mathcal{P} \big\}. \tag{14}$$

If $\mathcal{V}'_k$ is empty, the algorithm stops with the assurance that $u_k = C_{\mathrm{L}}$. Indeed, if $\mathbf{v}_{k,1}, \ldots, \mathbf{v}_{k,m_k} \in \mathcal{P}$, then

$$C_{\mathrm{L}} = \min_{\mathbf{r}} \max_{\mathbf{p} \in \mathcal{P}} D(\mathbf{p} || \mathbf{r}) \geq \min_{\mathbf{r}} \max_{j \in \{1, \ldots, m_k\}} D(\mathbf{v}_{k,j} || \mathbf{r}) = u_k.$$

But since u_k is an upper bound on C_{L}, equality holds. If $\mathcal{V}'_k$ is nonempty, we construct a new polytope

$$\mathcal{P}_{k+1} = \mathcal{P}_k \cap \tilde{\mathcal{P}}_k, \tag{15}$$

where $\tilde{\mathcal{P}}_k$ is the intersection of the half-spaces corresponding to the hyperplanes of Corollary 3, i.e.,

$$\tilde{\mathcal{P}}_k = \bigcap_{\mathbf{v} \in \mathcal{V}'_k} \left\{ \mathbf{p} : \left\langle \mathbf{p}, \log \frac{\mathbf{v}}{\mathbf{r}_k} \right\rangle \leq \max_{s \in \mathcal{S}} \left\langle \mathbf{w}(s), \log \frac{\mathbf{v}}{\mathbf{r}_k} \right\rangle \right\}.$$

Going from $\mathcal{P}_k$ to $\mathcal{P}_{k+1}$, we are "cutting off" the points in $\mathcal{V}'_k$ from $\mathcal{P}_k$. The k-th polytope may thus be written as

$$\mathcal{P}_k = \mathcal{P}_0 \cap \bigcap_{\kappa=0}^{k-1} \tilde{\mathcal{P}}_\kappa.$$

The body of the main algorithm can now be summarized as follows.

Initialization: Initialize with $\mathcal{P}_0$ as in (7).
Step k: Compute $\mathcal{V}_k$ from $\mathcal{P}_k$. From $\mathcal{V}_k$ compute u_k and $\mathbf{q}_k$ and determine the vertices $\mathbf{v}_{k,1}, \ldots, \mathbf{v}_{k,m_k}$ (using Blahut-Arimoto or otherwise). Test the vertices $\mathbf{v}_{k,1}, \ldots, \mathbf{v}_{k,m_k}$ for membership in $\mathcal{P}$ to obtain $\mathcal{V}'_k$. If $\mathcal{V}'_k = \emptyset$, declare "$C_{\mathrm{L}} = u_k$" and stop. Otherwise, compute $\mathcal{P}_{k+1}$ and go to step $k+1$.

Theorem 1 (Convergence of the Upper Bounds)**.** *If the algorithm terminates after the k-th iteration, i.e., if $\mathcal{V}'_k = \emptyset$, then $u_k = C_{\mathrm{L}}$. If the algorithm does not terminate after a finite number of steps, i.e., if $\mathcal{V}'_k \neq \emptyset$ for all k, then $u_k \downarrow C_{\mathrm{L}}$ as $k \to \infty$.*

Proof: The first statement has already been proved. Assume therefore that $\mathcal{V}'_k$ is nonempty for all k. Since $\{u_k\}_{k=0}^{\infty}$ is nonincreasing and lower bounded by C_{L}, we have $u_k \downarrow u_\star$ for some $u_\star \geq C_{\mathrm{L}}$. It remains to show that $u_\star \leq C_{\mathrm{L}}$.

Since, by (8), $m_k \in \{2, \ldots, |\mathcal{Y}|\}$ for all k, there exists $m \in \{2, \ldots, |\mathcal{Y}|\}$ and a strictly increasing sequence of positive integers $\{\lambda_k\}_{k=0}^{\infty}$ such that $m_{\lambda_k} = m$ for all k. Consider the sequence of tuples

$$(\mathbf{v}_{\lambda_k,1}, \ldots, \mathbf{v}_{\lambda_k,m}, \mathbf{q}_{\lambda_k}), \quad k = 0, 1, \ldots \tag{16}$$

and recall from (9) that

$$\mathbf{r}_{\lambda_k} = \sum_{j=1}^{m} q_{\lambda_k}^{(j)} \mathbf{v}_{\lambda_k,j}.$$

Since the sequence (16) is from a compact set, there exists a strictly increasing sequence of positive integers $\{\mu_k\}_{k=0}^{\infty}$ and a tuple

$$(\mathbf{v}_{\star,1}, \ldots, \mathbf{v}_{\star,m}, \mathbf{q}_\star),$$

where $\mathbf{q}_\star$ is a probability vector in $\mathbb{R}^m$, and where $\mathbf{v}_{\star,j} \in \mathcal{P}_0$ for $j = 1, \ldots, m$, such that

$$(\mathbf{v}_{\lambda_{\mu_k},1}, \ldots, \mathbf{v}_{\lambda_{\mu_k},m}, \mathbf{q}_{\lambda_{\mu_k}}) \to (\mathbf{v}_{\star,1}, \ldots, \mathbf{v}_{\star,m}, \mathbf{q}_\star),$$

as $k \to \infty$. Define

$$\mathbf{r}_\star \triangleq \sum_{j=1}^{m} q_\star^{(j)} \mathbf{v}_{\star,j}.$$

Then

$$\mathbf{r}_{\lambda_{\mu_k}} \to \mathbf{r}_\star, \quad (k \to \infty).$$

For convenience, put $\nu_k = \lambda_{\mu_k}$. We will show that $\mathbf{v}_{\star,j} \in \mathcal{P}$ for $j = 1, \ldots, m$. To this end, we prove the inequalities

$$D(\mathbf{v}_{\star,j} || \mathbf{r}_\star) \leq \max_{s \in \mathcal{S}} \left\langle \mathbf{w}(s), \log \frac{\mathbf{v}_{\star,j}}{\mathbf{r}_\star} \right\rangle, \tag{17}$$

and

$$D(\mathbf{v}_{\star,j}||\mathbf{r}_\star) \geq \max_{s\in\mathcal{S}} D(\mathbf{w}(s)||\mathbf{r}_\star). \tag{18}$$

To show (17), observe that for all k

$$\left\langle \mathbf{v}_{\nu_{k+1},j}, \log \frac{\mathbf{v}_{\nu_k,j}}{\mathbf{r}_{\nu_k}} \right\rangle \leq \max_{s\in\mathcal{S}} \left\langle \mathbf{w}(s), \log \frac{\mathbf{v}_{\nu_k,j}}{\mathbf{r}_{\nu_k}} \right\rangle. \tag{19}$$

Indeed, if $\mathbf{v}_{\nu_k,j} \notin \mathcal{P}$, then one of the constraints defining $\mathcal{P}_{\nu_{k+1}}$ is

$$\left\langle \mathbf{p}, \log \frac{\mathbf{v}_{\nu_k,j}}{\mathbf{r}_{\nu_k}} \right\rangle \leq \max_{s\in\mathcal{S}} \left\langle \mathbf{w}(s), \log \frac{\mathbf{v}_{\nu_k,j}}{\mathbf{r}_{\nu_k}} \right\rangle, \quad \text{for all } \mathbf{p},$$

and since $\mathbf{v}_{\nu_{k+1},j}$ is in $\mathcal{P}_{\nu_{k+1}}$, (19) holds. If $\mathbf{v}_{\nu_k,j} \in \mathcal{P}$, then, by Corollary 2,

$$D(\mathbf{v}_{\nu_k,j}||\mathbf{r}_{\nu_k}) = \max_{s\in\mathcal{S}} \left\langle \mathbf{w}(s), \log \frac{\mathbf{v}_{\nu_k,j}}{\mathbf{r}_{\nu_k}} \right\rangle. \tag{20}$$

But since $\mathbf{v}_{\nu_{k+1},j} \in \mathcal{P}_{\nu_{k+1}} \subset \mathcal{P}_{\nu_k}$, it follows from Proposition 4 that

$$\left\langle \mathbf{v}_{\nu_{k+1},j}, \log \frac{\mathbf{v}_{\nu_k,j}}{\mathbf{r}_{\nu_k}} \right\rangle \leq D(\mathbf{v}_{\nu_k,j}||\mathbf{r}_{\nu_k}). \tag{21}$$

Combining (20) and (21) we conclude that (19) holds. Taking $k \to \infty$ on both sides of (19) we obtain (17). To prove (18), observe that

$$\begin{aligned} D(\mathbf{v}_{\nu_k,j}||\mathbf{r}_{\nu_k}) &= \max_{\mathbf{p}\in\mathcal{P}_{\nu_k}} D(\mathbf{p}||\mathbf{r}_{\nu_k}) \\ &\geq \max_{\mathbf{p}\in\mathcal{P}} D(\mathbf{p}||\mathbf{r}_{\nu_k}) \\ &= \max_{s\in\mathcal{S}} D(\mathbf{w}(s)||\mathbf{r}_{\nu_k}), \end{aligned}$$

where we used (12) in the first line, the fact that $\mathcal{P} \subseteq \mathcal{P}_{\nu_k}$ in the second line, and the convexity of relative entropy in the last line. Letting $k \to \infty$ gives (18). Combining (17) and (18) and using Proposition 3, we obtain

$$\begin{aligned} D(\mathbf{v}_{\star,j}||\mathbf{r}_\star) &\leq \max_{s\in\mathcal{S}} \left\langle \mathbf{w}(s), \log \frac{\mathbf{v}_{\star,j}}{\mathbf{r}_\star} \right\rangle \\ &\leq \max_{s\in\mathcal{S}} D(\mathbf{w}(s)||\mathbf{r}_\star) \\ &\leq D(\mathbf{v}_{\star,j}||\mathbf{r}_\star). \end{aligned}$$

Consequently,

$$\max_{s\in\mathcal{S}} \left\langle \mathbf{w}(s), \log \frac{\mathbf{v}_{\star,j}}{\mathbf{r}_\star} \right\rangle = \max_{s\in\mathcal{S}} D(\mathbf{w}(s)||\mathbf{r}_\star),$$

which, by Proposition 3, implies that $\mathbf{v}_{\star,j} = \mathbf{w}(s_j^\star)$ for some $s_j^\star \in \mathcal{S}$. We conclude that $\mathbf{v}_{\star,j} \in \mathcal{P}$ for $j = 1, \ldots, m$. Thus,

$$\begin{aligned} C_{\mathrm{L}} &\geq H\left(\sum_{j=1}^m q_\star^{(j)} \mathbf{v}_{\star,j}\right) - \sum_{j=1}^m q_\star^{(j)} H(\mathbf{v}_{\star,j}) \\ &= \lim_{k\to\infty} \left(H(\mathbf{r}_{\nu_k}) - \sum_{j=1}^m q_{\nu_k}^{(j)} H(\mathbf{v}_{\nu_k,j}) \right) \\ &= \lim_{k\to\infty} u_{\nu_k} \\ &= u_\star, \end{aligned}$$

where, in the second line, we used the continuity of entropy (see [13, p. 33, Lemma 2.7]). ∎

We now extend the main algorithm so that it produce an additional monotonic sequence $\{l_k\}_{k=0}^\infty$ of lower bounds. This sequence will be shown to converge to C_{L} after at most a finite number of steps.

For m_k as in (8) and $j = 1, \ldots, m_k$ let $s_{k,j}^\star \in \mathcal{S}$ be a solution of the maximization

$$\max_{s\in\mathcal{S}} \left\langle \mathbf{w}(s), \log \frac{\mathbf{v}_{k,j}}{\mathbf{r}_k} \right\rangle, \tag{22}$$

which is carried out by the main algorithm to determine membership of $\mathbf{v}_{k,j}$ in $\mathcal{P}$ (see Corollary 2). Define the sets

$$\mathcal{W}_k \triangleq \bigcup_{\kappa=0}^k \bigcup_{j=1}^{m_\kappa} \{\mathbf{w}(s_{\kappa,j}^\star)\}, \quad k = 0, 1, \ldots$$

Since $\mathcal{W}_k \subseteq \mathcal{W}$, running a capacity algorithm on $\mathcal{W}_k$ yields a lower bound l_k on C_{L}:

$$l_k \triangleq \min_{\mathbf{r}} \max_{\mathbf{w}\in\mathcal{W}_k} D(\mathbf{w}||\mathbf{r}), \quad k = 0, 1, \ldots$$

The l_k's are nondecreasing because $\mathcal{W}_k \subseteq \mathcal{W}_{k+1}$. Note that the convex hull of $\mathcal{W}_k$ is an inner approximation of $\mathcal{P}$.

Theorem 2 (Convergence of the Lower Bounds)**.** *If the algorithm stops after the k-th iteration, i.e., if $\mathcal{V}_k' = \emptyset$, then $l_k = C_{\mathrm{L}}$. If the algorithm does not stop after a finite number of steps, i.e., if $\mathcal{V}_k' \neq \emptyset$ for all k, then $l_k = C_{\mathrm{L}}$ for all sufficiently large k.*

Proof: If $\mathcal{V}_k' = \emptyset$, then for all $j \in \{1, \ldots, m_k\}$ we have $\mathbf{v}_{k,j} = \mathbf{w}(s_j)$ for some $s_j \in \mathcal{S}$. It suffices to show that $\mathbf{w}(s_j) \in \mathcal{W}_k$ for all $j \in \{1, \ldots, m_k\}$. If $\mathbf{w}(s) \neq \mathbf{w}(s_j)$, then

$$\begin{aligned} \left\langle \mathbf{w}(s), \log \frac{\mathbf{v}_{k,j}}{\mathbf{r}_k} \right\rangle &= \left\langle \mathbf{w}(s), \log \frac{\mathbf{w}(s_j)}{\mathbf{r}_k} \right\rangle \\ &< D(\mathbf{w}(s)||\mathbf{r}_k) \\ &\leq u_k \\ &= D(\mathbf{v}_{k,j}||\mathbf{r}_k) \\ &= \left\langle \mathbf{w}(s_j), \log \frac{\mathbf{v}_{k,j}}{\mathbf{r}_k} \right\rangle, \end{aligned} \tag{23}$$

where we used Proposition 3 in the second line, (11) and the fact that $\mathcal{P} \subseteq \mathcal{P}_k$ in the third line, (10) in the fourth line, and the fact that $\mathbf{v}_{k,j} = \mathbf{w}(s_j)$ in the last line. Since the inequality is strict, it follows that $\mathbf{w}(s_{k,j}^\star) = \mathbf{w}(s_j)$, and we conclude that $\mathbf{w}(s_j) \in \mathcal{W}_k$ for all $j \in \{1, \ldots, m_k\}$.

If the algorithm does not terminate after a finite number of steps, consider for every $j \in \{1, \ldots, m\}$ the limit $\mathbf{v}_{\star,j}$ of the subsequence $\{\mathbf{v}_{\nu_k,j}\}_{k=0}^\infty$ as in the proof of Theorem 1. We show that $\mathbf{v}_{\star,j} \in \mathcal{W}_k$ for all sufficiently large k. From the proof of Theorem 1 we know that for all $j \in \{1, \ldots, m\}$ we have $\mathbf{v}_{\star,j} = \mathbf{w}(s_j^\star)$ for some $s_j^\star \in \mathcal{S}$. Thus, for every $s \in \mathcal{S}$ such that $\mathbf{w}(s) \neq \mathbf{w}(s_j^\star)$,

$$\begin{aligned} \left\langle \mathbf{w}(s), \log \frac{\mathbf{v}_{\star,j}}{\mathbf{r}_\star} \right\rangle &= \left\langle \mathbf{w}(s), \log \frac{\mathbf{w}(s_j^\star)}{\mathbf{r}_\star} \right\rangle \\ &< D(\mathbf{w}(s)||\mathbf{r}_\star), \end{aligned} \tag{24}$$

by Proposition 3. Moreover, for every k,

$$D(\mathbf{w}(s)||\mathbf{r}_{\nu_k}) \leq u_{\nu_k},$$

by (11) and the fact that $\mathcal{P} \subseteq \mathcal{P}_{\nu_k}$. Taking $k \to \infty$, we obtain

$$D(\mathbf{w}(s)||\mathbf{r}_\star) \leq u_\star. \tag{25}$$

We further have

$$\begin{aligned} D(\mathbf{v}_{\star,j}||\mathbf{r}_\star) &= \lim_{k\to\infty} D(\mathbf{v}_{\nu_k,j}||\mathbf{r}_{\nu_k}) \\ &= \lim_{k\to\infty} u_{\nu_k} \\ &= u_\star. \end{aligned} \tag{26}$$

where we used (10) in the second line. Combining (24), (25), and (26) gives

$$\left\langle \mathbf{w}(s), \log \frac{\mathbf{v}_{\star,j}}{\mathbf{r}_\star} \right\rangle < D(\mathbf{v}_{\star,j}||\mathbf{r}_\star), \quad \mathbf{w}(s) \neq \mathbf{w}(s_j^\star).$$

Thus, for every $s \in \mathcal{S}$ satisfying $\mathbf{w}(s) \neq \mathbf{w}(s_j^\star)$, we have that

$$\epsilon(s) \triangleq D(\mathbf{v}_{\star,j}||\mathbf{r}_\star) - \left\langle \mathbf{w}(s), \log \frac{\mathbf{v}_{\star,j}}{\mathbf{r}_\star} \right\rangle \tag{27}$$

is positive. By continuity, for every $s \in \mathcal{S}$, there is $\Delta(k,s) \geq 0$ such that

$$\begin{aligned} &\left\langle \mathbf{w}(s_j^\star), \log \frac{\mathbf{v}_{\nu_k,j}}{\mathbf{r}_{\nu_k}} \right\rangle - \left\langle \mathbf{w}(s), \log \frac{\mathbf{v}_{\nu_k,j}}{\mathbf{r}_{\nu_k}} \right\rangle \\ &\geq \left\langle \mathbf{w}(s_j^\star), \log \frac{\mathbf{v}_{\star,j}}{\mathbf{r}_\star} \right\rangle - \left\langle \mathbf{w}(s), \log \frac{\mathbf{v}_{\star,j}}{\mathbf{r}_\star} \right\rangle - \Delta(k,s), \end{aligned} \tag{28}$$

where $\Delta(k,s) \to 0$ as $k \to \infty$. Combining (27), (28), and the fact that $\mathbf{v}_{\star,j} = \mathbf{w}(s_j^\star)$, we obtain for every $s \in \mathcal{S}$ such that $\mathbf{w}(s) \neq \mathbf{w}(s_j^\star)$,

$$\begin{aligned} &\left\langle \mathbf{w}(s_j^\star), \log \frac{\mathbf{v}_{\nu_k,j}}{\mathbf{r}_{\nu_k}} \right\rangle - \left\langle \mathbf{w}(s), \log \frac{\mathbf{v}_{\nu_k,j}}{\mathbf{r}_{\nu_k}} \right\rangle \\ &\geq \epsilon(s) - \Delta(k,s) \\ &\geq \min_{\substack{s' \in \mathcal{S}: \\ \mathbf{w}(s') \neq \mathbf{w}(s_j^\star)}} \big(\epsilon(s') - \Delta(k,s')\big) \\ &> 0, \end{aligned}$$

for all large enough k. It follows that for every $j \in \{1,\ldots,m\}$ there exists $k(j)$ such that $\mathbf{w}(s^\star_{\nu_{k(j)},j}) = \mathbf{w}(s_j^\star)$. We conclude that $\{\mathbf{v}_{\star,1},\ldots,\mathbf{v}_{\star,m}\} \subseteq \mathcal{W}_k$, and hence $l_k = C_{\mathrm{L}}$, for all sufficiently large k. ∎

V. Implementation and Performance

We have not yet addressed the question of how to compute the vertices of the polytopes $\mathcal{P}_k$. There is extensive literature on the subject, and a number of approaches can be found in [12, Chapter II, Sec. 4.2]. A typical implementation is based on "pivot operations", as used in the Simplex method [15]. It should be noted that it is not necessary to compute all the vertices of $\mathcal{P}_k$ in every iteration. Indeed, since $\mathcal{P}_k$ is the intersection of $\mathcal{P}_{k-1}$ with a number of half-spaces, one has to compute only the new vertices generated at the intersections of $\mathcal{P}_{k-1}$ with the hyperplanes corresponding to the half-spaces, and discard all vertices of $\mathcal{P}_{k-1}$ that have been "cut off".

Another important question is how the algorithm scales with the problem size. Unfortunately, we do not have a complexity analysis. Our simulations suggest, however, that the performance depends heavily on $|\mathcal{Y}|$, and only mildly on $|\mathcal{X}|$ and L. For example, for $|\mathcal{Y}| = 4$ we were able to compute C_{L} to within 10^{-5} of the exact value in a few seconds. For $|\mathcal{Y}| = 7$ this took several hours. We therefore predict that the usefulness of the algorithm is limited to small values of $|\mathcal{Y}|$.

We noticed that most of the computation time goes into computing l_k, u_k and $\mathbf{q}_k$; in this setting, it seems that the Cutting-Plane algorithm outperforms the Blahut-Arimoto algorithm. We also noticed that the lower bounds l_k typically give a good approximation of C_{L} after just a few iterations.

Acknowledgment

The authors would like to thank Ligong Wang for helpful discussions.

References

[1] L. Lastras-Montano, M. Franceschini, T. Mittelholzer, and M. Sharma, "Rewritable storage channels," *Proc. ISITA 2008*, pp. 7–10.

[2] T. Mittelholzer, M. Franceschini, L. Lastras-Montano, I. Elfadel, and M. Sharma, "Rewritable channels with data-dependent noise," in *2009 International Conference on Communications*, 2009, pp. 1–6.

[3] M. Franceschini, L. Lastras-Montano, T. Mittelholzer, and M. Sharma, "The role of feedback in rewritable storage channels [Lecture Notes," *IEEE Signal Processing Magazine*, vol. 26, pp. 190–194, 2009.

[4] L. Lastras-Montano, T. Mittelholzer, and M. Franceschini, "Superposition coding in rewritable channels," in *Information Theory and Applications workshop (ITA2010)*.

[5] L. Lastras-Montano, M. Franceschini, and T. Mittelholzer, "The capacity of the uniform noise rewritable channel with average cost," in *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 201–205.

[6] T. Mittelholzer, L. Lastras-Montano, M. Sharma, and M. Franceschini, "Rewritable storage channels with limited number of rewrite iterations," in *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 973–977.

[7] C. Bunte and A. Lapidoth, "On the storage capacity of rewritable memories," in *Electrical and Electronics Engineers in Israel (IEEEI), 2010 IEEE 26th Convention of*. IEEE, pp. 402–405.

[8] R. Blahut, "Computation of channel capacity and rate-distortion functions," *Information Theory, IEEE Transactions on*, vol. 18, no. 4, pp. 460–473, 2002.

[9] S. Arimoto, "An algorithm for computing the capacity of arbitrary discrete memoryless channels," *Information Theory, IEEE Transactions on*, vol. 18, no. 1, pp. 14–20, 2002.

[10] X. Liang, "A fast algorithm for computing the capacity of discrete memoryless channels," in *Information Sciences and Systems (CISS), 2010 44th Annual Conference on*. IEEE, pp. 1–6.

[11] R. Gallager, *Information theory and reliable communication*. John Wiley & Sons, Inc. New York, NY, USA, 1968.

[12] R. Horst and H. Tuy, *Global optimization: Deterministic approaches*. Springer Verlag, 1996.

[13] I. Csiszár and J. Körner, *Information theory: coding theorems for discrete memoryless systems*. Academic Press, Inc. Orlando, FL, USA, 1982.

[14] J. Huang and S. Meyn, "Characterization and computation of optimal distributions for channel coding," *Information Theory, IEEE Transactions on*, vol. 51, no. 7, pp. 2336–2351, 2005.

[15] D. Bertsimas and J. Tsitsiklis, *Introduction to linear optimization*. Athena Scientific Belmont, MA, 1997.