Bounds on the Rate of 2-D Bit-Stuffing Encoders*

Ido Tal Ron M. Roth
Computer Science Department
Technion, Haifa 32000, Israel.
Email: idotal@ieee.org, ronny@cs.technion.ac.il

***Abstract*—A method for bounding the rate of bit-stuffing encoders for 2-D constraints is presented. Instead of considering the original encoder, we consider a related one which is quasi-stationary. We use the quasi-stationary property in order to formulate linear requirements that must hold on the probabilities of the constrained arrays that are generated by the encoder. These requirements are used as part of a linear program. The minimum and maximum of the linear program bound the rate of the encoder from below and from above, respectively.**

A lower bound on the rate of an encoder is also a lower bound on the capacity of the corresponding constraint. For some constraints, our results lead to tighter lower bounds than what was previously known.

***Index Terms*—Bit-stuffing encoders, Linear programming, Runlength-limited constraints, Two-dimensional constraints, Quasi-stationary distribution.**

I. Introduction

Two-dimensional (2-D) constraints are formally defined in [1]. Consider a 2-D constraint $\mathbb{S}$ defined over some finite alphabet Σ. Informally, a bit-stuffing encoder for $\mathbb{S}$ operates as follows. We encode information to an $M \times N$ rectangular array; namely, we produce an array $a \in \mathbb{S} \cap \Sigma^{M \times N}$. We first initialize the "boundaries" of the array (formally defined later) according to some fixed probability distribution. Then, we write to the "interior" of the array in raster fashion: row-by-row. The symbol currently written is the result of a coin toss. The probability distribution of the coin is a function of neighboring symbols, which have already been written. However, the "coins" used are in fact (invertible) probability transformers, the input of which is the information we wish to encode. Thus, information can be encoded, and decoded.

A bit-stuffing encoder is "variable-rate". The bit-stuffing technique was initially devised for encoding one-dimensional (1-D) constraints [2]. In [3] and [4], bit-stuffing encoders for specific 2-D constraints were presented and analyzed. In [5], a slightly different definition of bit-stuffing was used to give lower bounds on the capacity of specific 2-D constraints.

In this work, we derive upper and lower bounds on the rate of a general bit-stuffing encoder. A lower bound on the rate of an encoder is also a lower bound on the capacity of the corresponding constraint:

$$\mathsf{cap}(\mathbb{S}) = \lim_{M,N\to\infty} \frac{1}{M \cdot N} \cdot \log_2 \left|\mathbb{S} \cap \Sigma^{M \times N}\right| .$$

For some constraints, our results lead to tighter lower bounds on capacity than what was previously known.

$$\begin{array}{cccccccc}
1 & \mathbf{0} & 0 & 0 & \mathbf{0} & 1 & \mathbf{0} & 0 \\
\mathbf{0} & \mathbf{0} & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
0 & 0 & 0 & 0 & 0 & \mathbf{0} & 1 & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & 1 & \mathbf{0} & 0 & 0 & 0 & 0 & 0
\end{array}$$

Fig. 1. Binary array satisfying the non-attacking-kings constraint. If we flip any one (or more) of the highlighted "0" bits to "1", then the resulting array will not satisfy the non-attacking-kings constraint.

Fix some 2-D constraint $\mathbb{S}$ over an alphabet Σ. As a running example, consider the non-attacking-kings constraint, also called the square constraint, $\mathbb{S}_{\mathrm{sq}}$, defined over the binary alphabet $\Sigma_{\mathrm{sq}} = \{0, 1\}$ (see Figure 1). A binary array satisfies the non-attacking-kings constraint if each entry set to "1" has all of its eight-neighbors set to "0". Namely, two entries equal to "1" may not appear consecutively along a row, column, or diagonal.

The rest of this paper is organized as follows. In Sections II and III, we define our notation and our model of a bit-stuffing encoder, respectively. In Section IV, we define the concept of quasi-stationarity. We also prove that, w.l.o.g., we may assume that our encoder is quasi-stationary. In Section V we take this a step further: we consider a relatively small array with a corresponding probability distribution which is (truly) stationary. The stationarity trait is modeled as part of a linear program. The minimum (maximum) of the linear program bounds the rate of our encoder from below (above). Finally, section VI states a generic lower bound on capacity, and contains examples where this bound improves on previous results.

We note at this point that although this work deals with 2-D constraints, our method can be easily generalized to higher dimensions as well.

II. Notation

We first set up some notation.

Parallelogram and rectangle: For $M, N > 0$ and $t \geq 0$, denote

$$\mathsf{B}^{(t)}_{M,N} = \{(i, j) : 0 \leq i < M \ , \quad 0 \leq t \cdot i + j < N\} \ .$$

* This work was supported by grant No. 2002197 from the United-States–Israel Binational Science Foundation (BSF), Jerusalem, Israel. The results of this work were presented at the IEEE International Symposium on Information Theory, Toronto, Ontario, Canada, July 2008.

I. Tal is with the Center for Magnetic Recording Research (CMRR), University of California at San Diego, La Jolla, CA 92093-0401, USA. This work was done while he was with the Computer Science Department, Technion, Haifa 32000, Israel (email: idotal@ieee.org).
R. M. Roth is with the Computer Science Department, Technion, Haifa 32000, Israel (email: ronny@cs.technion.ac.il).

Also, for $t = 0$, denote

$$\mathsf{B}_{M,N} = \mathsf{B}^{(0)}_{M,N} \ .$$

Configuration: Let $a = (a_{i,j})_{(i,j)\in\mathsf{U}}$ be a 2-D configuration over Σ. Namely, the index set satisfies $\mathsf{U} \subseteq \mathbb{Z}^2$, and for all $(i,j) \in \mathsf{U}$ we have that $a_{i,j} \in \Sigma$.

Shifts: For integers α, β we denote the shifted index set as

$$\sigma_{\alpha,\beta}(\mathsf{U}) = \{(i+\alpha, j+\beta) : (i,j) \in \mathsf{U}\} \ .$$

Also, by abuse of notation, let $\sigma_{\alpha,\beta}(a)$ be the shifted configuration (with index set $\sigma(\mathsf{U})$):

$$\sigma_{\alpha,\beta}(a)_{i+\alpha,j+\beta} = a_{i,j} \ .$$

Restriction of configuration: For an index set $\Psi \subseteq \mathsf{U}$, denote the restriction of a to Ψ by $a[\Psi] = (a[\Psi]_{i,j})_{(i,j)\in\Psi}$. Namely,

$$a[\Psi]_{i,j} = a_{i,j} \ , \quad \text{where} \quad (i,j) \in \Psi \ .$$

Shift and restrict: Let $\tau_{\alpha,\beta}(a,\Psi)$ be shorthand for

$$\tau_{\alpha,\beta}(a,\Psi) = (\sigma_{-\alpha,-\beta}(a))[\Psi] \ .$$

Namely, shift the configuration a such that index (α,β) is now index $(0,0)$, and then restrict to Ψ.

Boundary: Denote by $\partial(\mathsf{U},\Psi)$ the set of all the indexes $(\alpha,\beta) \in \mathsf{U}$ for which the "shift and restrict" operation is invalid.

$$\partial(\mathsf{U},\Psi) = \{(\alpha,\beta) \in \mathsf{U} : \sigma_{\alpha,\beta}(\Psi) \not\subseteq \mathsf{U}\} \ .$$

The index set $\partial(\mathsf{U},\Psi)$ is termed the "boundary", and the "interior" is

$$\bar{\partial}(\mathsf{U},\Psi) = \mathsf{U} \setminus \partial(\mathsf{U},\Psi) \ .$$

When $\mathsf{U} = \mathsf{B}_{M,N}$ and Ψ is understood from the context, we abbreviate

$$\partial_{M,N} = \partial(\mathsf{B}_{M,N},\Psi) \ , \quad \bar{\partial}_{M,N} = \bar{\partial}(\mathsf{B}_{M,N},\Psi) \ .$$

Figure 2 shows an example of such sets, where

$$\Psi = \{(0,-2),(0,-1),(-1,-1),(-1,0),(-1,1)\} \ . \quad (1)$$

Restriction of constraint: Denote the restriction of $\mathbb{S}$ to U by

$$\mathbb{S}[\mathsf{U}] = \{a : \text{there exists } a' \in \mathbb{S} \text{ such that } a'[\mathsf{U}] = a\} \ .$$

If $\mathsf{U} = \mathsf{B}_{M,N}$, then we abbreviate

$$\mathbb{S}_{M,N} = \mathbb{S}[\mathsf{B}_{M,N}] \ .$$

Lexicographic ordering: We define a lexicographic ordering $\prec$ on $\mathbb{Z}^2$ as

$$(i',j') \prec (i,j) \quad \Longleftrightarrow \quad (i' < i) \ \text{or} \ (i' = i \quad \text{and} \quad j' < j) \ .$$

Also, we define the index set

$$\mathsf{T}_{i,j} = \{(i',j') : (i',j') \prec (i,j)\} \ . \quad (2)$$

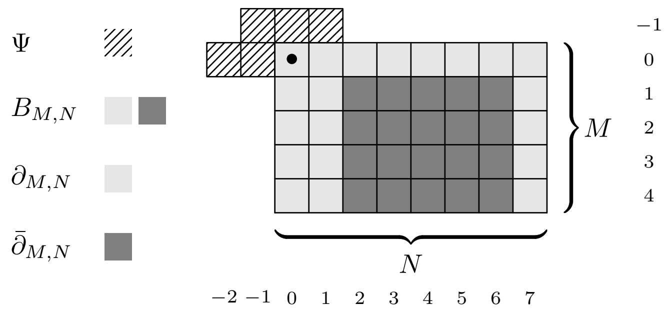

Fig. 2. The index $(0,0)$ is represented by $\bullet$. We take Ψ as in (1), and it is represented by the diagonally striped cells. We set $M = 5$ and $N = 8$. The index set $\mathsf{B}_{M,N}$ is represented by the shaded part (both light and dark). The boundary $\partial_{M,N}$ is represented by the lighter shaded part, while the interior $\bar{\partial}_{M,N}$ is represented by the darker shaded part.

$$\varphi^{(0)} = \begin{array}{ccccc} & 0 & 0 & 0 \\ 0 & 0 & \bullet & \end{array} \qquad \varphi^{(1)} = \begin{array}{ccccc} & 0 & 0 & 0 \\ 1 & 0 & \bullet & \end{array}$$

Fig. 3. The two non-trivial configurations for μ in our running example, where $\bullet$ designates coordinate $(0,0)$.

III. Bit stuffer definitions

In this section, we present the formal definition of bit-stuffing encoders. A bit-stuffing encoder for $\mathbb{S}$ is defined through a triple

$$\mathcal{E} = (\Psi, \mu, \boldsymbol{\delta} = (\delta_{M,N})_{M,N>0}) \ .$$

The set

$$\Psi \subseteq \mathsf{T}_{0,0} \quad (3)$$

is termed the *neighbor set*. The *conditional probability function* μ,

$$\mu(\cdot|\cdot) \ , \quad \mu : \Sigma \times \mathbb{S}[\Psi] \to [0,1] \ ,$$

is a conditional probability distribution on Σ, given an element of $\mathbb{S}[\Psi]$. For $M, N > 0$, the *boundary probability function*

$$\delta_{M,N} : \mathbb{S}[\partial_{M,N}] \to [0,1]$$

is a probability distribution on $\mathbb{S}[\partial_{M,N}]$. From here onward, we fix $\mathcal{E}$.

For our running example, let the neighbor set $\Psi_{\text{sq}} = \Psi$ be as in (1), and define $\varphi^{(0)}, \varphi^{(1)} \in \mathbb{S}_{\text{sq}}[\Psi]$ as

$$\begin{array}{lllll} \varphi^{(0)}_{0,-2}=0 & \varphi^{(0)}_{0,-1}=0 & \varphi^{(0)}_{-1,-1}=0 & \varphi^{(0)}_{-1,0}=0 & \varphi^{(0)}_{-1,1}=0 \\ \varphi^{(1)}_{0,-2}=1 & \varphi^{(1)}_{0,-1}=0 & \varphi^{(1)}_{-1,-1}=0 & \varphi^{(1)}_{-1,0}=0 & \varphi^{(1)}_{-1,1}=0 \end{array}$$

(see Figure 3). Also, take the conditional probability function as

$$\mu_{\text{sq}}(1|\varphi) = 1 - \mu_{\text{sq}}(0|\varphi) = \begin{cases} 0.258132 & \varphi = \varphi^{(0)} \\ 0.312231 & \varphi = \varphi^{(1)} \\ 0 & \text{otherwise} \ . \end{cases} \quad (4)$$

Thus, $\mu_{\text{sq}}(\cdot|\cdot)$ can be implemented using two coins (one for the context $\varphi^{(0)}$ and one for $\varphi^{(1)}$). For our running example, we take $\delta_{M,N}$ as the function equal to 1 for the all zero boundary $(0)_{(i,j)\in\partial_{M,N}}$, and 0 for all other members of $\mathbb{S}_{\text{sq}}[\partial_{M,N}]$.

Given integers $M, N > 0$, the bit-stuffing encoder $\mathcal{E}$ defines a probability measure on the elements $a = (a_{i,j})_{(i,j) \in \mathsf{B}_{M,N}}$ of $\mathsf{B}_{M,N}$, in the following manner. As a first step, we set the boundary $a[\partial_{M,N}]$, according to the probability distribution $\delta_{M,N}$. Next, we write the contents of the interior of a in raster fashion: row-by-row, from left to right. The probability of writing $w \in \Sigma$ in entry $(i,j) \in \bar{\partial}_{M,N}$ is given by

$$\mathrm{Prob}(a_{i,j} = w) = \mu(w|(\tau_{i,j}(a, \Psi)) .$$

Specifically, note that when writing entry (i,j), we have by (3) that $\tau_{i,j}(a)$ is a function of entries of a which have already been written. A fundamental requirement for Ψ and μ is that for every M, N, and $\delta_{M,N}$, the support of the probability measure thus defined is contained in $\mathbb{S}_{M,N}$.

Let

$$A(\mathcal{E}, M, N) = A = (A_{i,j})_{(i,j) \in \mathsf{B}_{M,N}}$$

be a random variable taking values on $\mathbb{S}_{M,N}$ according to the measure we have just defined. Namely,

$$\mathrm{Prob}(A = a) = \delta_{M,N}(a[\partial_{M,N}]) \cdot \prod_{(i,j) \in \bar{\partial}_{M,N}} \mu(a_{i,j} | \tau_{i,j}(a, \Psi)) . \quad (5)$$

We now explain how $\mathcal{E}$ is used to actually encode information. The "coin tosses" corresponding to the invocations of μ are, in effect, a function of the information we wish to encode. Specifically, the values of the tosses are the output of distribution transformers on the input stream (the mapping from the input stream to the sequence of coin tosses is one-to-one) [4]. Thus, we may encode information, and also decode it. So, we define the *rate* of our encoder as

$$R(\mathcal{E}) \triangleq \liminf_{M,N \to \infty} \frac{H(A[\bar{\partial}_{M,N}] | A[\partial_{M,N}])}{M \cdot N} ,$$

where

$$A = A(\mathcal{E}, M, N) .$$

Note that since

$$\liminf_{M,N \to \infty} \frac{|\bar{\partial}_{M,N}|}{M \cdot N} = 1 ,$$

we also have that

$$R(\mathcal{E}) = \liminf_{M,N \to \infty} \frac{H(A(\mathcal{E}, M, N))}{M \cdot N} .$$

IV. Quasi-stationarity

This section is devoted to defining the concept of quasi-stationarity [3, §6]. Note that the probability distribution corresponding to the output induced by $\mathcal{E}$ need not, in general, be stationary. However, as we show next, we can assume w.l.o.g. that it is — in a sense — locally close to stationary.

Fix $k > 0$. Define the random variable

$$A^{(k)}(\mathcal{E}, M, N) = A^{(k)} = (A^{(k)}_{i,j})_{(i,j) \in \mathsf{B}_{M,N}}$$

taking values on $\mathbb{S}_{M,N}$ as follows. For $w \in \mathbb{S}_{M,N}$, we have

$$\mathrm{Prob}(A^{(k)} = w) = \frac{1}{k^2} \sum_{0 \le i,j < k} \mathrm{Prob}(\sigma_{-i,-j}(A'[\mathsf{B}_{M,N}]) = w) ,$$

where

$$A' = A(\mathcal{E}, M + k - 1, N + k - 1) .$$

Namely, given A', we randomly and uniformly pick an $M \times N$ sub-configuration of it, and shift accordingly. The usefulness of $A^{(k)}$ is that it is "quasi-stationary".

Lemma 1 ([3, Proposition 6.1]): Let $\mathcal{E}$, M, N, and k be given. Let $\mathsf{U} \subseteq \mathsf{B}_{M,N}$ be an index set, and let $w \in \mathbb{S}[\mathsf{U}]$ be given. Suppose that for given integers α, β we have that $\sigma_{\alpha,\beta}(\mathsf{U}) \subseteq \mathsf{B}_{M,N}$. Denote $A^{(k)} = A^{(k)}(\mathcal{E}, M, N)$. Then,

$$\left| \mathrm{Prob}(A^{(k)}[\mathsf{U}] = w) - \mathrm{Prob}(A^{(k)}[\sigma_{\alpha,\beta}(\mathsf{U})] = \sigma_{\alpha,\beta}(w)) \right| \le \frac{|\alpha| + |\beta|}{k} .$$

Next, we show that $A^{(k)}$ is a random variable corresponding to an encoder very similar to $\mathcal{E}$. First, define $\boldsymbol{\delta}^{(k)} = (\delta^{(k)}_{M,N})_{M,N>0}$, where

$$\delta^{(k)}_{M,N} : \mathbb{S}[\partial_{M,N}] \to [0,1]$$

(that is, $\delta^{(k)}_{M,N}$ is a probability distribution on $\mathbb{S}[\partial_{M,N}]$), and for every $d \in \mathbb{S}[\partial_{M,N}]$,

$$\delta^{(k)}_{M,N}(d) = \mathrm{Prob}(A^{(k)}(\mathcal{E}, M, N)[\partial_{M,N}] = d) .$$

Next, define the encoder $\mathcal{E}^{(k)}$ as

$$\mathcal{E}^{(k)} = (\Psi, \mu, \boldsymbol{\delta}^{(k)}) . \quad (6)$$

Lemma 2 ([3, Proposition 6.2]): The probability distributions of $A^{(k)}(\mathcal{E}, M, N)$ and $A(\mathcal{E}^{(k)}, M, N)$ are equal.

The next lemma essentially states that the normalized entropies of A and $A^{(k)}$ are asymptotically equal (for $M, N \to \infty$ and k fixed). The proof is straightforward.

Lemma 3: Fix an integer $k > 0$. Then,

$$R(\mathcal{E}) = R(\mathcal{E}^{(k)}) .$$

It follows from Lemma 3 that we can obtain bounds on $R(\mathcal{E})$ by bounding instead the rate of the quasi-stationary encoder $\mathcal{E}^{(k)}$. And, indeed, quasi-stationarity will turn out to be useful for this purpose.

V. Linear program

In this section, we present lower and upper bounds on $R(\mathcal{E})$. The bounds will be expressed as values of corresponding linear programs. In a nutshell, we will effectively assume that each $r \times s$ sub-array has a corresponding distribution which is stationary; this assumption will be translated into equations in the linear programs. The lower (upper) bound will result from the worst (best) such distribution.

For $r, s > 0$ and $t \ge 0$, we say that the parallelogram $\mathsf{B}^{(t)}_{r,s}$ is valid with respect to the neighbor set Ψ if the set

$$\left\{ (\alpha, \beta) : (\Psi \cup (0,0)) \subseteq \sigma_{\alpha,\beta}(\mathsf{B}^{(t)}_{r,s}) \right\} \quad (7)$$

is non-empty. Namely, some shift of the parallelogram includes the neighbor set Ψ and $(0,0)$. From here onward, we fix r, s, and t so that $\mathsf{B}^{(t)}_{r,s}$ is valid. Also, we fix u and v, where (u,v) is the largest element of (7), with respect to the ordering $\prec$.

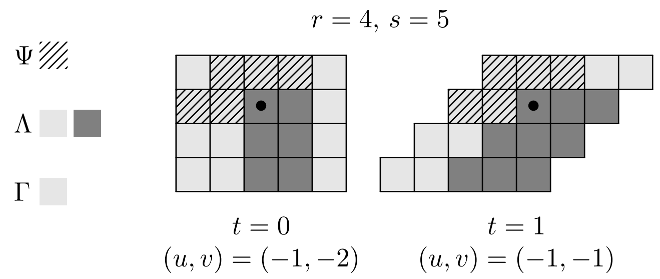

Fig. 4. The index sets Ψ, Λ, and Γ. The index sets are shown for $r = 4$, $s = 5$, and for both $t = 0$ and $t = 1$. The index $(0,0)$ is represented by $\bullet$. We take Ψ as in (1), and it is represented by the diagonally striped cells. The index set Λ is represented by the shaded part (both light and dark). The boundary Γ is represented by the lighter shaded part. Note that $\Psi \subseteq \Gamma \subseteq \Lambda$.

Denote (see Figure 4)

$$\Lambda = \sigma_{u,v}(\mathsf{B}^{(t)}_{r,s}) \ , \quad \Gamma = \partial(\Lambda, \Psi) \ .$$

For an as yet unspecified probability distribution over $\mathbb{S}[\Gamma]$

$$\pi(z) \ , \quad z \in \mathbb{S}[\Gamma] \ ,$$

define the random variable Y taking values on $\mathbb{S}[\Lambda]$ as follows. For $y \in \mathbb{S}[\Lambda]$,

$$\mathrm{Prob}(Y = y) = \pi(y[\Gamma]) \prod_{(i,j)\in\Lambda\setminus\Gamma} \mu(y_{i,j} | \tau_{i,j}(y, \Psi)) \qquad (8)$$

(compare to (5)). Note that $\mathrm{Prob}(Y = y)$ is a linear function of the various $\pi(z)$'s. Next, define

$$\Lambda' = \sigma_{u,v}(\mathsf{B}^{(t)}_{r-1,s}) \ , \quad \Lambda'' = \sigma_{u,v}(\mathsf{B}^{(t)}_{r,s-1}) \ ,$$

and

$$\Gamma' = \partial(\Lambda', \Psi) \ , \quad \Gamma'' = \partial(\Lambda'', \Psi) \ .$$

Consider the linear program in Figure 5. First, note that it is indeed a linear program. Namely, recall that by (8), the probability distribution of Y is a linear function of the $\pi(z)$'s. Thus, both sides of (9) and (10) are also linear functions of the $\pi(z)$'s. For example, the LHS of (9) equals

$$\sum_{y\in\mathbb{S}[\Lambda]:y[\Gamma']=z'} \pi(y[\Gamma]) \prod_{(i,j)\in\Lambda\setminus\Gamma} \mu(y_{i,j} | \tau_{i,j}(y, \Psi)) \ .$$

Denote the value of the linear program when minimizing by $\mathrm{lp}^*_{\min} = \mathrm{lp}^*_{\min}(\mathcal{E})$, and when maximizing by $\mathrm{lp}^*_{\max} = \mathrm{lp}^*_{\max}(\mathcal{E})$. Since (5) and (8) are very similar, we may intuitively say that $\mathcal{E}$ outputs Y. The optimization is over the probability distribution of the boundary $Y[\Gamma]$. The linear requirements (9) and (10) are added to force the distribution of Y to be stationary. The objective function is the rate at point $(0,0)$.

The following theorem is our main result.

Theorem 4: For the linear program in Figure 5, we have that

$$\mathrm{lp}^*_{\min} \le R(\mathcal{E}) \le \mathrm{lp}^*_{\max} \ .$$

In order to prove the theorem, we first state and prove a lemma, on a slightly modified linear program.

Lemma 5: Fix $k > 0$, and replace (9) and (10) in Figure 5 by

$$\Big|\mathrm{Prob}(Y[\Gamma'] = z') - \mathrm{Prob}(Y[\sigma_{0,1}(\Gamma')] = \sigma_{0,1}(z'))\Big| \le \frac{1}{k}$$

and

$$\Big|\mathrm{Prob}(Y[\Gamma''] = z'') - \mathrm{Prob}(Y[\sigma_{1,-t}(\Gamma'')] = \sigma_{1,-t}(z''))\Big| \le \frac{t+1}{k} \ ,$$

respectively.

Denote the minimum and maximum of the resulting linear program as $\mathrm{lp}^{(k)}_{\min}$ and $\mathrm{lp}^{(k)}_{\max}$, respectively. Then,

$$\mathrm{lp}^{(k)}_{\min} \le R(\mathcal{E}) \le \mathrm{lp}^{(k)}_{\max} \ .$$

Proof: Consider $\mathcal{E}^{(k)}$ (as defined by (6)). For given M and N, define the index sets

$$\mathsf{E} = \partial(\mathsf{B}_{M,N}, \Lambda) \qquad \text{and} \qquad \mathsf{I} = \bar{\partial}(\mathsf{B}_{M,N}, \Lambda) \ ,$$

where the letters stand for (on the) "edge" and "inside", respectively. Obviously,

$$\lim_{M,N\to\infty} \frac{|\mathsf{I}|}{M \cdot N} = 1 \ . \qquad (11)$$

Denote $A^{(k)} = A^{(k)}(\mathcal{E}, M, N)$. By (11) and Lemma 2,

$$R(\mathcal{E}^{(k)}) = \liminf_{M,N\to\infty} \frac{H(A^{(k)}[\mathsf{I}] | A^{(k)}[\mathsf{E}])}{|\mathsf{I}|} \ .$$

Minimize (Maximize)

$$-\sum_{z\in\mathbb{S}[\Gamma]} \pi(z) \sum_{w\in\Sigma} \mu(w|z[\Psi]) \log_2 \mu(w|z[\Psi])$$

over the variables $(\pi(z) : z \in \mathbb{S}[\Gamma])$, subject to the following:

$$\sum_{z\in\mathbb{S}[\Gamma]} \pi(z) = 1 \ .$$

For all $z \in \mathbb{S}[\Gamma]$,

$$\pi(z) \ge 0 \ .$$

For all $z' \in \mathbb{S}[\Gamma']$,

$$\mathrm{Prob}(Y[\Gamma'] = z') = \mathrm{Prob}(Y[\sigma_{0,1}(\Gamma')] = \sigma_{0,1}(z')) \ . \qquad (9)$$

For all $z'' \in \mathbb{S}[\Gamma'']$,

$$\mathrm{Prob}(Y[\Gamma''] = z'') = \mathrm{Prob}(Y[\sigma_{1,-t}(\Gamma'')] = \sigma_{1,-t}(z'')) \ . \qquad (10)$$

Fig. 5. Linear program. The minimum (maximum) value is denoted $\mathrm{lp}^*_{\min}$ ($\mathrm{lp}^*_{\max}$) and is a lower (upper) bound on $R(\mathcal{E})$.

Notice that $\Psi \subseteq \Lambda$. Thus, $\mathsf{I} \subseteq \bar{\partial}_{M,N}$, and we have

$$H(A^{(k)}[\mathsf{I}]|A^{(k)}[\mathsf{E}]) = \sum_{(i,j)\in\mathsf{I}} H(A^{(k)}_{i,j}|A^{(k)}[\mathsf{T}_{i,j} \cap \mathsf{B}_{M,N}])$$
$$= \sum_{(i,j)\in\mathsf{I}} H(A^{(k)}_{i,j}|\tau_{i,j}(A^{(k)}, \Psi)) ,$$

where $\mathsf{T}_{i,j}$ is as defined in (2) and the last equality follows from (5).

We now prove the following claim: for all $(i,j) \in \mathsf{I}$, we have that

$$\mathrm{lp}^{(k)}_{\min} \le H(A^{(k)}_{i,j}|\tau_{i,j}(A^{(k)}, \Psi)) . \tag{12}$$

To see this, fix some $(i,j) \in \mathsf{I}$, and define for all $z \in \mathbb{S}[\Gamma]$,

$$p^{(k)}(z) = \mathrm{Prob}(\tau_{i,j}(A^{(k)}, \Gamma) = z) .$$

Substituting $\pi(z) = p^{(k)}(z)$, the objective function in Figure 5 is equal to $H(A^{(k)}_{i,j}|\tau_{i,j}(A^{(k)}, \Psi))$. Also, notice that the probability distribution of Y is equal to that of $\tau_{i,j}(A^{(k)}, \Lambda)$. By the fact that $A^{(k)}$ is quasi-stationary (and thus, so is every sub-configuration of it), all the linear requirements in the modified linear program are satisfied (i.e., the $p^{(k)}(z)$'s form a feasible solution). So, our claim (12) is proved.

We conclude that $\mathrm{lp}^{(k)}_{\min} \le R(\mathcal{E}^{(k)})$. Thus, by Lemma 3,

$$\mathrm{lp}^{(k)}_{\min} \le R(\mathcal{E}) .$$

A similar proof yields $R(\mathcal{E}) \le \mathrm{lp}^{(k)}_{\max}$. ■

Proof of Theorem 4: First, note that the modified linear program defined in Lemma 5 has at least one feasible solution, $p^{(k)}(z)$, whenever M and N are large enough so that I is non-empty.

For a given k, denote the minimizing variable values of the modified linear program by $\pi^{(k)}(z)$, $z \in \mathbb{S}[\Gamma]$. Think of these variable values as a vector

$$\boldsymbol{\pi}^{(k)} = (\pi^{(k)}(z))_{z\in\mathbb{S}[\Gamma]} .$$

By compactness, the series $\boldsymbol{\pi}^{(k)}$, $k = 1, 2, \ldots$, has a cluster point, which we denote by $\boldsymbol{\pi}^*$. Obviously, $\boldsymbol{\pi}^*$ implies a feasible solution for the linear program in Figure 5. More so, we must also have that the value of the objective function for this feasible solution is a lower bound on $R(\mathcal{E})$. So,

$$\mathrm{lp}^*_{\min} \le R(\mathcal{E}) .$$

Similarly, we deduce that

$$R(\mathcal{E}) \le \mathrm{lp}^*_{\max} .$$

■

Remark: While the definition of the encoder $\mathcal{E}$ includes (besides Ψ and μ) also the boundary distributions $\boldsymbol{\delta} = (\delta_{M,N})_{M,N>0}$, the bounds $\mathrm{lp}^*_{\min}$ and $\mathrm{lp}^*_{\max}$ do *not* depend on $\boldsymbol{\delta}$.

Applying Theorem 4 to our running example, with $r = 4$, $s = 5$, $t = 1$, gives

$$0.42430953 \le R(\mathcal{E}) \le 0.42442765 .$$

TABLE I
BOUNDS ON THE RATES OF ENCODERS USING A SMALL NUMBER OF COINS.

Constraint	Coins	$\mathrm{lp}^*_{\min}$	$\mathrm{lp}^*_{\max}$	[3]
$(2,\infty)$-RLL	1	0.440722	0.444679	0.4267
$(3,\infty)$-RLL	1	0.349086	0.386584	0.3402
n.i.b.	2	0.917730	0.919395	0.9127
$(1,\infty)$-RLL	3	0.587776	0.587785	—

To the best of our knowledge, our running example is the highest rate bit-stuffing encoder known, given that we are allowed to use at most two coins (i.e., two probability transformers). For comparison, we have calculated by the method presented in [6] that

$$\mathsf{cap}(\mathbb{S}_{\mathrm{sq}}) \le 0.425078 .$$

Namely, with two coins we achieve a rate that is only 0.2% less than capacity.

Table I contains our results for a number of constraints (the full descriptions of the corresponding encoders are given in the Appendix). We abbreviate the "no isolated bits" constraints as "n.i.b.". In the first three rows, we compare ourselves to the results in [3] (Table 1 and Equation (12)). For the comparison to be fair, we restrict ourselves to the neighbor sets Ψ used in [3], and use the same number of coins.

VI. A LOWER BOUND ON CAPACITY

The following is a straightforward corollary of Theorem 4.

Corollary 6: For every bit-stuffing encoder $\mathcal{E}$,

$$\mathrm{lp}^*_{\min}(\mathcal{E}) \le \mathsf{cap}(\mathbb{S}) .$$

Thus, we can use the minimizing linear program of Figure 5 to bound $\mathsf{cap}(\mathbb{S})$ from below.

To obtain better lower bounds on $\mathsf{cap}(\mathbb{S})$, we can search for good Ψ and μ. For instance, for the set $\Psi = \Psi_{\mathrm{sq}}$ in (1), the function μ_{sq} in (4) was obtained by maximizing $\mathrm{lp}^*_{\min}$ over all μ that form with Ψ_{sq} (and every $\boldsymbol{\delta}$) a bit-stuffing encoder for $\mathbb{S}_{\mathrm{sq}}$. Better lower bounds can be obtained by looking at larger sets Ψ (at the price of higher computational complexity).

Table II summarizes our results for certain constraints (the Appendix contains the full descriptions of the encoder corresponding to the first row). The last two columns contain previously published lower bounds on the capacity of the corresponding constraint. We have highlighted values of $\mathrm{lp}^*_{\min}$ which are an improvement of these previously known results. The bounds in the penultimate column are taken from [7] (to appear as [8]), which was published recently. We note that the method used in [8] is quite different than ours. As can be seen, both [8] and our method are comparable. The bounds in the last column are taken from [9], [5], [10], and [11], respectively: they were the the best known when our method was first published in [12] (at the same time as [7]). We note that during the review process of this paper, the authors of [9] extended their method [13], thereby improving their lower bound for the $(2,\infty)$-RLL constraint to 0.4453.

TABLE II
BOUNDS ON THE RATES OF CERTAIN BIT-STUFFING ENCODERS.

Constraint	Coins	$\mathrm{lp}^*_{\min}$	$\mathrm{lp}^*_{\max}$	[7]	Others
$(2,\infty)$-RLL	5	**0.44420**	0.4450	0.44417	0.4423
$(3,\infty)$-RLL	2	0.35973	0.3690	0.36562	0.3641
$(0,2)$-RLL	66	0.81549	0.8169	0.81600	0.7736
	18	0.81501	0.8162		
	9	0.81073	0.8197		
n.i.b.	56	**0.92264**	0.9238	0.92086	0.9156

ACKNOWLEDGMENT

The first author wishes to thank Roee Engelberg for very stimulating discussions.

APPENDIX

Tables **??**–VI specify the encoders listed in Table I along with the parameters r, s, and t used when computing the bounds therein. Table VII provides the respective details for the encoder in the first row of Table II. The top row in Tables **??**–VII contains the number of coins in the encoder and the values of r, s, and t. All subsequent rows in each table are organized as follows: the left column contains the specification of non-trivial $\varphi^{(i)}$, as was done in Figure 3, and the right column is the value of $\mu(0|\varphi^{(i)})$ (of course, $\mu(1|\varphi^{(i)}) = 1 - \mu(0|\varphi^{(i)})$). Note that we do not specify $\boldsymbol{\delta}$, since the bounds do not depend on it. (The encoder in Table V is the same as in [3], except for slight modifications of the values of $\mu(0|\varphi^{(i)})$.)

TABLE III
ENCODER FOR THE $(2,\infty)$-RLL CONSTRAINT.

Coins $= 1$ $\quad r = 6 \quad s = 5 \quad t = 0$	
$\varphi^{(0)} = \begin{array}{ccc} & & 0 \\ & & 0 \\ 0 & 0 & \bullet \end{array}$	$\mu(0\vert\varphi^{(0)}) = 0.712079$

TABLE IV
ENCODER FOR THE $(3,\infty)$-RLL CONSTRAINT.

Coins $= 1$ $\quad r = 5 \quad s = 6 \quad t = 0$	
$\varphi^{(0)} = \begin{array}{cccc} & & & 0 \\ & & & 0 \\ & & & 0 \\ 0 & 0 & 0 & \bullet \end{array}$	$\mu(0\vert\varphi^{(0)}) = 0.767299$

TABLE V
ENCODER FOR THE N.I.B. CONSTRAINT.

Coins $= 2$ $\quad r = 4 \quad s = 4 \quad t = 1$
Defined as in [3], except for substituting
$\frac{2}{3} \Rightarrow 0.65 \qquad \frac{1}{2} \Rightarrow 0.53$

TABLE VI
ENCODER FOR THE $(1,\infty)$-RLL CONSTRAINT.

Coins $= 3$ $\quad r = 4 \quad s = 5 \quad t = 2$	
$\varphi^{(0)} = \begin{array}{cccc} & 0 & 0 & 0 \\ 0 & \bullet & & \end{array}$	$\mu(0\vert\varphi^{(0)}) = 0.65975$
$\varphi^{(1)} = \begin{array}{cccc} & 0 & 1 & 0 \\ 0 & \bullet & & \end{array}$	$\mu(0\vert\varphi^{(1)}) = 0.566761$
$\varphi^{(2)} = \begin{array}{cccc} & 0 & 0 & 1 \\ 0 & \bullet & & \end{array}$	$\mu(0\vert\varphi^{(2)}) = 0.700057$

TABLE VII
ADDITIONAL ENCODER FOR THE $(2,\infty)$-RLL CONSTRAINT.

Coins $= 5$ $\quad r = 5 \quad s = 5 \quad t = 1$	
$\varphi^{(0)} = \begin{array}{ccccc} & & 0 & 0 & 0 \\ & & 0 & 0 & \\ 0 & 0 & \bullet & & \end{array}$	$\mu(0\vert\varphi^{(0)}) = 0.736453$
$\varphi^{(1)} = \begin{array}{ccccc} & & 0 & 0 & 0 \\ & & 0 & 1 & \\ 0 & 0 & \bullet & & \end{array}$	$\mu(0\vert\varphi^{(1)}) = 0.681881$
$\varphi^{(2)} = \begin{array}{ccccc} & & 0 & 0 & 1 \\ & & 0 & 0 & \\ 0 & 0 & \bullet & & \end{array}$	$\mu(0\vert\varphi^{(2)}) = 0.697553$
$\varphi^{(3)} = \begin{array}{ccccc} & & 0 & 0 & 1 \\ & & 0 & 1 & \\ 0 & 0 & \bullet & & \end{array}$	$\mu(0\vert\varphi^{(3)}) = 0.611369$
$\varphi^{(4)} = \begin{array}{ccccc} & & 0 & 1 & 0 \\ & & 0 & 0 & \\ 0 & 0 & \bullet & & \end{array}$	$\mu(0\vert\varphi^{(4)}) = 0.664354$

REFERENCES

[1] S. Halevy and R. M. Roth, “Parallel constrained coding with application to two-dimensional constraints,” *IEEE Trans. Inform. Theory*, vol. 48, pp. 1009–1020, 2002.

[2] P. Bender and J. K. Wolf, “A universal algorithm for generating optimal and nearly optimal run-length-limited, charge constrained binary sequences,” in *Proc. IEEE Int'l Symp. Inform. Theory (ISIT'1993)*, San Antonio, Texas, 1993, p. 6.

[3] S. Halevy, J. Chen, R. M. Roth, P. H. Siegel, and J. K. Wolf, “Improved bit-stuffing bounds on two-dimensional constraints,” *IEEE Trans. Inform. Theory*, vol. 50, pp. 824–838, 2004.

[4] R. M. Roth, P. H. Siegel, and J. K. Wolf, “Efficient coding schemes for the Hard-Square model,” *IEEE Trans. Inform. Theory*, vol. 47, pp. 1166–1176, 2001.

[5] S. Forchhammer and T. V. Laursen, “Entropy of bit-stuffing-induced measures for two-dimensional checkerboard constraints,” *IEEE Trans. Inform. Theory*, vol. 53, pp. 1537–1546, 2007.

[6] N. Calkin and H. S. Wilf, “The number of independent sets in a grid graph,” *SIAM J. Discrete Math.*, vol. 11, pp. 54–60, 1997.

[7] A. Sharov and R. M. Roth, “Two-dimensional constrained coding based on tiling,” in *Proc. IEEE Int'l Symp. Inform. Theory (ISIT'2008)*, Toronto, Ontario, 2008, pp. 1468–1472.

[8] ——, “Two-dimensional constrained coding based on tiling,” *IEEE Trans. Inform. Theory*, to appear.

[9] E. Ordentlich and R. M. Roth, “Capacity lower bounds and approximate enumerative coding for 2-D constraints,” in *Proc. IEEE Int'l Symp. Inform. Theory (ISIT'2007)*, Nice, France, 2007, pp. 1681–1685.

[10] J. J. Ashley and B. H. Marcus, “Two-dimensional low-pass filtering codes,” *IEEE Trans. Commmun.*, vol. 46, pp. 724–727, 1998.

[11] S. Forchhammer and T. V. Laursen, “A model for the two-dimensional no isolated bits constraint,” in *Proc. IEEE Int'l Symp. Inform. Theory (ISIT'2006)*, Seattle, Washington, 2006, pp. 1189–1193.

[12] I. Tal and R. M. Roth, “Bounds on the rate of 2-D bit-stuffing encoders,” in *Proc. IEEE Int'l Symp. Inform. Theory (ISIT'2008)*, Toronto, Ontario, Canada, 2008.

[13] E. Ordentlich and R. M. Roth, “Approximate enumerative coding for 2-D constraints through ratios of matrix products,” in *Proc. IEEE Int'l Symp. Inform. Theory (ISIT'2009)*, Seoul, Korea, 2009, pp. 1050–1054.

Biographies:

Ido Tal was born in Haifa, Israel, in 1975. He received the B.Sc., M.Sc., and Ph.D. degrees in computer science from Technion—Israel Institute of Technology, Haifa, Israel, in 1998, 2003 and 2009, respectively.

He is a Postdoctoral Scholar at the Center for Magnetic Recording Research (CMRR), University of California at San Diego, La Jolla, CA, USA. His research interests include constrained coding and error-control coding.

Ron M. Roth was born in Ramat Gan, Israel, in 1958. He received the B.Sc. degree in computer engineering, the M.Sc. in electrical engineering and the D.Sc. in computer science from Technion—Israel Institute of Technology, Haifa, Israel, in 1980, 1984 and 1988, respectively. Since 1988 he has been with the Computer Science Department at Technion, where he now holds the General Yaakov Dori Chair in Engineering. During the academic years 1989–91 he was a Visiting Scientist at IBM Research Division, Almaden Research Center, San Jose, California, and during 1996–97 and 2004–05 he was on sabbatical leave at Hewlett–Packard Laboratories, Palo Alto, California. He is the author of the book *Introduction to Coding Theory*, published by Cambridge University Press in 2006. Dr. Roth was an associate editor for coding theory in IEEE TRANSACTIONS ON INFORMATION THEORY from 1998 till 2001, and he is now serving as an associate editor in *SIAM Journal on Discrete Mathematics*. His research interests include coding theory, information theory, and their application to the theory of complexity.