Representation of 2-D RLL Constraints

Hiroshi Kamabe
Department of Information Science, Gifu University,
1-1, Yanagido, Gifu, 501-1193 Japan.
Email: kamabe@ieee.org

***Abstract*— Weakly constrained codes were introduced for constructing efficient encoding algorithms with low complexities, that is, encoding algorithm which can be implemented with smaller circuits and less memories. In this paper we introduce 2 dimensional (2D) weak constraint codes for 2D storage media and propose some coding rules for 2D weak run-length-limited constraints. Simulation results show that these rules almost strictly obey given constraints. We can show these results theoretically for some constraints. The encoding algorithm also can be considered as a representation or a model of a 2D Run-Length-Limited(RLL) constraint. We show that the model of 2D constraints is complete for 2D (d, k) RLL constraint for positive integers d and with $2d \leq k$ and that the model is incomplete for a 2D $(2, 4)$ RLL constraints.**

Keyword: Bit-stuffing, RLL constraint, 2 dimensional constraints.

I. Introduction

In one dimensional (1D) digital recording systems, e.g., hard disk and CD, we assume that recording channels have some input constraints because there are some sequences which can not be recorded properly. Therefore we use recording codes which encode data sequences into sequences satisfying given input constraints for the recording system.

Recent high density magnetic recording disk systems employ the perpendicular recording scheme because the the recording density of perpendicular recording systems is higher than that of traditional longitude recording systems. However, 2 dimensional(2D) recording systems, e.g., hologram recording systems and bit patterned media, are now studied extensively in order to achieve the higher recording density, e.g., 2 Tera-Bit/inch^2. Therefore several authors have studied encoding algorithms and capacities of 2D input constraints[1], [2], [3], [4], [5]. It is known that 2D input constraints are very different from 1D input constraints. For example, although we can apply a well developed mathematical theory in determining the capacity and constructing encoders for a certain class of 1D constraints (constrains modeled as sofic shifts)[6], [7], there is almost no general mathematical theory which can be applied to most of two dimensional input constraints. Furthermore, although there is a simple and useful representation of a 1D input constraint and we can describe 2D input constraints by specifying properties to be satisfied, we do not have any universal 'operational' representations for 2D input constraints yet.

The complexities of an encoder and a decoder are very important in digital recording system because only very small amount of energy can be consumed by them and the space for them is very limited in practical or commercial recording systems. By using the concept of 'weak constraint' we can construct efficient encoders with low complexity [8] for 1D constraints. We employ the concept of 'weak constraint codes' in designing an encoding algorithm for a 2D RLL constraint.

In the first part of this paper we show encoding rules for encoding a 1-D data sequence into a 2D pattern satisfying a given 2D RLL constraint. We show that these rules violate the given constraints but the degree of violation is not so large[1]. Then we give a general description of horizontal bit stuffing encoders (algorithms) with local maps, which can be regarded as encoders for 2D weak input constrains. The encoders can also be regarded as representations of 2D input constraints. We show that the encoders are complete for 2D $(1, k)$ RLL constraints and incomplete for a $(2, k)$ RLL constraint with $2 \leq d$ and $2d \leq k$. We also show that encoders implemented by a local map without looking forward are incomplete even for 2D $(1, k)$ constraints.

II. Run Length Limited Constraints

We consider channels for communication or storage with alphabet $\{-, +\}$ and assume that d and k are positive integers with $1 \leq d < k$. A 1-D (d, k) RLL constraint is defined as follows: the run length of a symbol is at least d and at most k where the run length is the length of a subsequence consisting of the same symbol. A 1D (d, k) limited constraint limits lengths of runs of symbol 0's between adjacent 1's: it requires that the length of the run of symbol 0 should be at least d and at most k. Both 1D (d, k) limited and 1D (d, k) RLL constraints are extended to constraints on 2D binary arrays (or planes). We consider 2D arrays in which $+$ and $-$ are located at lattice points. We assume that d_1, d_2, k_1 and k_2 are positive integers with $1 \leq d_1 < k_1$ and $1 \leq d_2 < k_2$. We say that a 2D pattern satisfies a 2D $(d_1, k_1; d_2, k_2)$ RLL constraint if the length of the vertical run of a symbols is at least d_1 and at most k_1, and the length of the horizontal run of a symbols is at least d_2 and at most k_2. We write '(d, k) RLL constraint' to mean a 2D $(d, k; d, k)$ RLL constraint for short. Since 1D (d, k) limited and 1D (d, k) RLL constraints are almost identical [9, Chapter 4] with few exceptions, only (d, k) constraints are studied in 1D cases. But 2D (d, k) limited sequences and 2D (d, k) RLL sequences are different [10]. We study 2D (d, k) RLL constraints in this paper.

[1] A part of this work was presented at ITW2006, Chendu, China.

The capacity C_Σ of a 2D constraint Σ is defined as follows,

$$C_\Sigma = \lim_{\substack{m\to\infty \\ n\to\infty}} \frac{\log_2 N^\Sigma_{m,n}}{mn}, \tag{1}$$

where $N^\Sigma_{m,n}$ is the number of 2D patterns which satisfy the 2D constraint Σ and whose sizes are $m \times n$.

If a 2D pattern satisfies a 2D (d,k) RLL constraints almost everywhere but violates the constraint with a acceptable low probability(or at quite few positions), we say that the 2D pattern satisfies a 2D weak (d,k) RLL constraint.

For 1-D constraints, we have to construct a code which always satisfies a given constraint in many cases. However, there are cases where we can construct an efficient and simple encoder under the condition that we are allowed to weaken or violate the given constraint [8], [11]. On the other hand, in 2D cases, it is very hard to construct efficient and practical encoding schemes by which we can translate data sequences into 2D patterns satisfying the given 2D constraint strictly. In this paper, therefore, we give encoding schemes which produce patterns satisfying weak 2D constraints. Furthermore we can regard the encoders for 2D weak constraints as representations of the 2D constraints. Therefore we investigate the completeness of the encoders.

III. Weak 2-D $(1,k;1,k)$ RLL constraints

First we give encoding algorithms for 2D $(1,k)$-RLL constraints and then show how these algorithms violate the constraints.

Let k be a positive integer with $k \geq 2$. Let x_i, $i = 0, 1, \ldots$ be a binary input sequence to be translated into a 2D pattern. We assume that the size of the resulting 2D pattern should be $N \times M$, that is, the pattern should have N rows and M columns. By $y_{j,\ell}$ we mean a bit at the j-th row and ℓ-th column in the pattern.

We assume that we are given a 2D $(1,k)$ RLL constraint for a positive integer k with $k \geq 2$.

1) The first line $y_{1,\ell}$, $\ell = 1, \ldots, N$ are obtained by some encoding algorithm for the 1-D $(1,k)$ RLL constraint, which can be constructed by using a well developed coding theory for 1-D input constraints of finite type [7]. Let q be the number of data bits encoded into the first line.
2) We put $q \leftarrow q+1$. (In each iteration of Step 4) data bit x_q will be encoded.)
3) We introduce a variable $v(\ell)$, $\ell = 1, \ldots, M$ and we put $v(\ell) \leftarrow 1$ for $\ell = 1, \ldots, M$. (In the following steps, the ℓ-th value of v, $v(\ell)$, contains the length of a vertical run consisting of symbols produced in the previous steps and terminating at position $(j-1,\ell)$ when we are determining a channel bit at position (j,ℓ).)
4) We put $j \leftarrow 2$.
   a) If $v(1) \geq k$ then $y_{j,1} \leftarrow \overline{y}_{j-1,1}$. If $v(1) < k$ then $y_{j,1} \leftarrow x_q$ and $q \leftarrow q+1$.
   b) If $y_{j-1,1} = y_{j,1}$ then $v(1) \leftarrow v(1)+1$. If $y_{j-1,1} \neq y_{j,1}$ then $v(1) \leftarrow 1$.
   c) We put $\ell \leftarrow 1$ and $p \leftarrow 1$.
      i) We put $v(\ell) \leftarrow v(\ell)+1$, $O_v \leftarrow v(\ell) - k$, $p \leftarrow p+1$ and $O_h \leftarrow p - k$.
      ii) If $O_v > 0$ and $O_h > 0$ and $y_{j-1,\ell} = y_{j,\ell-1}$ then we put $y_{j,\ell} \leftarrow \overline{y}_{j,\ell-1}$, $p \leftarrow 1$ and $v(\ell) \leftarrow 1$ and go to step vii. .
      iii) If $O_v > 0$ and $O_h > 0$ and $y_{j,\ell-1} \neq y_{j-1,\ell}$ then we have the two cases:
         A) if $O_v \geq O_h$ then we put $y_{j,\ell} \leftarrow \overline{y}_{j-1,\ell}$ and $v(\ell) \leftarrow 1$
         B) if $O_v < O_h$ then we put $y_{j,\ell} \leftarrow \overline{y}_{j,\ell-1}$ and $p \leftarrow 1$.

         We go to step vii. .
      iv) If $O_v > 0$ and $O_h \leq 0$ then we put $y_{j,\ell} \leftarrow \overline{y}_{j-1,\ell}$ and $v(\ell) \leftarrow 1$. If $y_{j,\ell-1} \neq y_{j,\ell}$ then we put $p \leftarrow 1$. Go to step vii. .
      v) If $O_v \leq 0$ and $O_h > 0$ then we put $y_{j,\ell} \leftarrow \overline{y}_{j,\ell-1}$ and $p \leftarrow 1$. If $y_{j,\ell} \neq y_{j-1,\ell}$ then we put $v(\ell) \leftarrow 1$. Go to step vii. .
      vi) If $O_v \leq 0$ and $O_h \leq 0$ then we put $y_{j,\ell} \leftarrow x_q$ and $q \leftarrow q+1$. If $y_{j,\ell} \neq y_{j-1,\ell}$ then $v(\ell) \leftarrow 1$. If $y_{j,\ell} \neq y_{j,\ell-1}$ then $p \leftarrow 1$.
      vii) We put $\ell \leftarrow \ell+1$. If $\ell \leq M$ then we go to step i. .
   d) We put $j \leftarrow j+1$. If $j \leq N$ then we go to step (a).

Algorithm 1

A deadlock is one of major difficulties in constructing 2D patterns satisfying a 2D (d,k) RLL constraint by a greedy method like Algorithm 1, where by the deadlock we mean that each symbol violates the given vertical or horizontal constraint at a certain position. A deadlock might happen if we would encode data bits without taking into account the possibility of deadlocks in future steps. Our criterion for encoding here is that we encode data bits without looking future steps in advance and we violate the vertical or horizontal constraints if we encounter a deadlock. However, even in the case of deadlock we put a symbol so that none of given constraints would be violated infinitely.

We can show that

Proposition 1: Every 2D pattern produced by Algorithm 1 satisfies that every vertical and horizontal runs have length at most $k+1$.

Remark 1: In the above proposition we could say the 2D pattern satisfies a 2D $(1,k+1)$-RLL constraint. However, from the following proof we can see that the horizontal k constraint is violated only when we encounter a local configuration such as a

$$\begin{array}{c} \left.\begin{array}{c} \overline{a} \\ \overline{a} \\ \vdots \\ \overline{a} \end{array}\right\} k \\ \underbrace{a\ a\ \cdots\ a}_{k+1} \end{array}$$

where $a \in \{+,-\}$, $\overline{a} = +$ if $a = 1$, and $\overline{a} = -$ if $a = +$. We expect that the probability that the pattern happens is relatively small if k is not small. This is confirmed by computer simulations. Therefore we can say the resulting 2D pattern satisfies a weak 2D $(1,k)$-RLL constraint. Actually we have Theorem 1 in the end of this section. $\square$

Proof: We index positions of a 2D pattern as shown in Fig. 1. Note that this is the way of indexing an element(or point) in a matrix, not in a usual 2 Dimensional plane. We

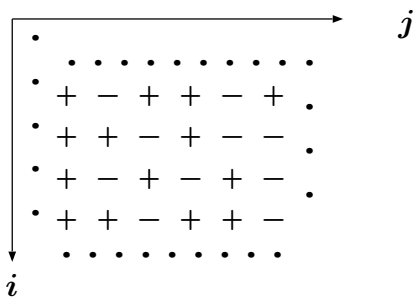

Fig. 1. Indices of positions

assume that we are now determining a symbol at the position (i,j). The values of $v(\ell)$ and p at the position represents the length of the vertical and horizontal run, respectively, terminating at the position. We mean the pair of $v(\ell)$ and p by $[v(\ell), p]$.

Suppose that Algorithm 1 produces a horizontal run of length $k+2$. Then we have the following configuration of v-p pairs

$$[A_{-2}, k] \quad [A_{-1}, k+1] \quad [A_0, k+2]$$

where $[A_0, k+2]$ is the v-p pair at (i,j), and so on. Since we have $[A_{-1}, k+1]$ at $(i-1,j)$, our algorithm must execute step 4)-c)-iii)-A) when it determines the symbol at $(i-1,j)$. This implies that the v-p pair at $(i-1, j-1)$ must be $[k, B_{-1}]$ for some integer B_{-1} and we must have the following configuration for v-p pairs.

$$\begin{array}{ccc} & [k, B_{-1}] & \\ [A_{-2}, k] & [A_{-1}, k+1] & [A_0, k+2] \end{array}$$

Since we have the following configuration

$$[A_{-1}, k+1] \quad [A_0, k+2]$$

our algorithm must execute 4)-c)-iii)-A) when determine the symbol at (i,j). This implies that the value of v-p pair at $(i, j-1)$ must be $[k+1, B_0]$ for some integer B_0 and finally we have the following

$$\begin{array}{ccc} & [k, B_{-1}] & [k+1, B_0] \\ [A_{-2}, k] & [A_{-1}, k+1] & [A_0, k+2] \end{array}$$

Suppose that the symbol at (i,j) is '+.' Then the above configuration corresponds to the following local pattern of channel symbols

$$\begin{array}{ccc} & - & - \\ & - & - \\ + & + & + \end{array}$$

When our algorithm determine the symbol at $(i, j-1)$ it must execute step 4)-c)-ii) or 4)-c)-iv). Hence, the symbol at $(i, j-1)$ must be '+' but we assumed the symbol is '−' in the above configuration. This is a contradiction. Therefore, we can conclude that every horizontal run length must be at most $k+1$.

Suppose that there is a vertical run of length $k+2$. Then the last symbol of the run must be generated in step 4)-c)-iii)-B) and there must be a horizontal run of length $\geq k+2$ terminating at $(i, j-1)$ to produce the vertical run of length $k+1$ in the step. However, in the first part of the proof we have shown that there is no horizontal run of length $k+2$. $\blacksquare$

Remark 2: Since violations in the above proof can happen only when special patterns appear. We wrote a program which implements our algorithm and ran the program under the following condition: $M = 30000, N = 30000, k = 4$. Let F_{vert} be the frequency of violation of the vertical constraint, that is, the ratio of the number of violations of the vertical constraint and the number of encoded bits. We define $F_{horizon}$ to be the frequency of violation of the horizontal constraint in the same way. When $\Pr\{x_q = +\} = \Pr\{x_q = 1\} = 0.5$, we found that $F_{vert} = 0.00196$ and $F_{horizon} = 4.38 \times 10^{-5}$. $\square$

Remark 3: The decoding function of Algorithm 1 is of finite scope, that is, when we decode a data bit at position (j, ℓ), we need only bits at positions $(j-i)$, $i = 0, 1, \ldots, k$ and $(j, \ell - i)$, $i = 0, 1, \ldots, k+1$. $\square$

If we modify Algorithm 1 as follows then resulting algorithm, say Algorithm 1': in step 4)-c)-i) we define O_v and O_h by

$$\begin{aligned} O_v &\leftarrow v(\ell) - k_1, \\ O_h &\leftarrow p - k_2. \end{aligned}$$

This algorithm produces 2D patterns satisfying a weak $(1, k_1; 1, k_2)$ constraint, that is,

1) the length of every vertical run is at most k_1,
2) the length of any horizontal run is at most $k_2 + 1$.

For a position (i,j), we define $P_v(i,j)$ to be the probability that the given 2D $(1,k)$-RLL constraint is violated when we employ the Algorithm 1.

Theorem 1: Let $1 < k$ be an integer and let (i,j) be a position. Suppose that the input sequence x_i, $i = 0, 1, \ldots$ is a bernoulli process with $\Pr(x_i = +) = \Pr(x_i = -) = 1/2$ and we employ the Algorithm 1. Then we have

$$P_v(i,j) \leq 2^{-2k}.$$

IV. Weak 2D $(d, k; 1, k)$ RLL constraints

Here we consider a coding rule for a $(d, k; 1, k)$ RLL constraint such that the vertical d RLL constraint is not violated but the vertical and horizontal k constraints may be violated. This implies that if $y_{j,\ell} \neq y_{j+1,\ell}$ then we must have $y_{j+1,\ell} = y_{j+2,\ell} = \cdots = y_{j+d,\ell}$. We say that positions (q, ℓ), $q = j+2, \ldots, j+d$ are in a d constrained segment if $y_{j,\ell} \neq y_{j+1,\ell}$.

We propose Algorithm 2 by adding the following two steps 4)-c)-i') and 4)-c)-i") before step 4)-c)-i) in Algorithm 1.

4)-c)-i') let m be a largest positive integer such that $(j, \ell + q)$ is in a d constrained segment and $y_{j-1,\ell+q} = y_{j-1,\ell+1}$ for $q = 1, \ldots, m$. If $p + m > k$ and $y_{j,\ell-1} = y_{j-1,\ell+1}$ then we put $y_{j,\ell} \leftarrow \overline{y}_{j,\ell+1}$ and go to step 4)-c)-vii).

4)-c)-i") if $y_{j-s,\ell} \neq y_{j-s+1,\ell}$ for some s with $2 \leq s \leq d$ then we put $y_{j,\ell} = y_{j-1,\ell}$ and go to step 4)-c)-vii).

We also replace step 4)-a) with the following step

4)-a') If $v(1) < d$ then $y_{j,1} \leftarrow y_{j-1,1}$. If $v(1) \geq k$ then $y_{j,1} \leftarrow \overline{y}_{j-1,1}$. If $d \leq v(1) < k$ then $y_{j,1} \leftarrow x_q$ and $q \leftarrow q + 1$.

Step 4)-c)-i') corresponds to the case where we are just one step before a long sequence of d constrained segment and step 4)-c)-i") corresponds to the case where we are just in a d constrained segment.

We can show that

Proposition 2: Let d and k be positive integers with $d < k$. Assume that a data sequence is encoded by using Algorithm 2. Then the resulting 2D pattern satisfies the following:

1) every vertical run has length at least d,
2) every vertical run has length at most $k + d - 1$,
3) every horizontal run has length at most $k + 2$.

Proof: The essential differences between Algorithm 1 and Algorithm 2 are step 4)-c)-i') and 4)-c)-i") given above. From step 4)-c)-i") we can conclude that the length of every vertical run is at least d.

Step 4)-c)-i') may violate the vertical k constraint because a channel bit at the present position is determined without taking account of run lengths including the bit at the position. Consider the following pattern

$$\begin{array}{c} \\ x_1 \\ \vdots \\ x_{d-1} \end{array} \underbrace{\begin{array}{cccc} + & + & \cdots & + \\ + & + & \cdots & + \\ \vdots & \vdots & \cdots & \vdots \\ + & + & \cdots & + \end{array}}_{k} \Bigg\} d$$

In this pattern $x_1, x_2, \ldots, x_{d-1}$ must be '$-$' from step 4)-c)-i'). This means the step may produce vertical runs of length $d - 1$. Therefore we can conclude that the $k \times d$ pattern of symbol '$+$' may extend the vertical runs by $d - 1$ symbols.

Step 4)-c)-iii) may also violate the horizontal k constraint and the vertical k constraint. We can see that violations of the given constraint happen in only these two steps. Therefore we concentrate on these steps in this proof.

We suppose that our second algorithm produces horizontal runs of length $k + 3$ and consider the first run $\boldsymbol{a} = y_{j,\ell-k-2} y_{j,\ell-k-1} \cdots y_{j,\ell}$ among those runs. We define $\boldsymbol{a}' = y_{j,\ell-k-2} \cdots y_{j,\ell-1}$ by deleting $y_{j,\ell}$ from $\boldsymbol{a}$.

The algorithm can produces $\boldsymbol{a}'$ in step 4)-c)-i') or 4)-c)-iii)-A). First we suppose that the algorithm produces run $\boldsymbol{a}'$ in 4)-c)-iii)-A), that is, $y_{j,\ell-1}$ is determined by step 4)-c)-iii)-A) and it happens that $y_{j,\ell-1} = y_{j,\ell-2}$. Since we have $O_v \geq O_h > 0$ at position $(j, \ell - 1)$, there is a vertical run $\boldsymbol{b}$ consisting of symbols at positions from $(j - k - 1, \ell - 1)$ to $(j - 1, \ell - 1)$ of length $k + 2$ with $y_{j,\ell-2} = \overline{y}_{j-1,\ell-1}$. Suppse that $y_{j,\ell-2}$ is determined by step 4)-c)-i'). Then we must have $y_{j,\ell-2} = \overline{y}_{j,\ell-1}$ because position $(j, \ell-1)$ is in a d constrained segment. But this contradicts our assumption on the length of $\boldsymbol{a}'$. Hence $y_{j,\ell-2}$ is determined by step 4)-c)-iii) and we have $y_{j-1,\ell-2} = \overline{y}_{j,\ell-2}$ and $y_{j-1,\ell-1} = \overline{y}_{j,\ell-1}$. This means $y_{j-1,\ell-2} = y_{j-1,\ell-1}$ and channel bit $y_{j-1,\ell-1}$ is determined in step 4)-c)-i'). Hence position $(j - 1, \ell)$ must be in a d constrained segment and we must have $y_{j-1,\ell} = \overline{y}_{j-1,\ell-1} = y_{j,\ell-1}$. Thus our algorithm let $\overline{y}_{j,\ell} = y_{j,\ell-1}$ in step 4)-c)-v). Therefore we can conclude that there is no horizontal run of length $k + 3$. An example of this case is shown as follows

$$\begin{array}{ccccc} & & & - & * \\ & & & - & * \\ & & - & - & + \\ & & - & - & + \\ + & + & + & + & + \end{array}$$

where $d = 2$ and $k = 3$. A horizontal run of length 5 at the bottom is the longest horizontal run. From the above argument we see that the symbol at the bottom-rightmost position must be '$-$.'

Next we prove that the length of every vertical run is at most $k + d$. As we have shown above step 4)-c)-i') may extend a vertical run by $d-1$ bits. From the definition of a d constrained segment we can see that there is no extension of a vertical run by the step just after another extension of the run by the step. Since vertical runs and horizontal runs of length k are allowed, there may be a horizontal run of $k + 1$ by ste 4)-c)-iii)-A) and, then, a vertical run of length $k + 1$ by step 4)-c)-iii)-B). Therefore we show that the length of vertical run without any violation by step 4)-c)-i') is at most $k + 1$.

A remaining case is an extension of a vertical run by a horizontal run $\boldsymbol{a}$ of length $k + 2$ in step 4)-c)-iii)-B). We remember that we have show $y_{j,\ell} = y_{j-1,\ell+1}$ in the first part of this proof. This means that $\boldsymbol{a}$ does not extend any vertical run in step 4)-c)-iii)-B). ■

Remark 4: By computer simulation we found that the horizontal and vertical k constraints are violated with probabilities 3.45×10^{-7} and 4.27×10^{-6}, respectively when $M = N = 30000$, $d = 4$, $k = 10$, and $\Pr\{x_q = +\} = \Pr\{x_q = 1\} = 0.5$.

V. Representations of Constraints

Input constraints of channels come from physical, engineering or economical constraints in many cases. Therefore most of them are originally stated in terms of physics. Fortunately, almost all 1D input constraints of practical interest have representations with finite directed graphs. The representations are also useful in designing encoders and decoders for the constraints. Unfortunately, we do not have any general method to represent 2D RLL constraints which also can be used in designing encoders, yet. In the remaining of this paper we, therefore, consider the representation problems of 2D RLL constraints. First first we define an encoder or encoding algorithm for a 2D RLL constraint and then we show that

the completeness and incompleteness of the encoder as a representation for the constraint.

Our encoding algorithms given in this paper could be considered as variants of bit stuffing algorithms introduced [12], [13] for several 2D input constraints. A bit stuffing algorithm can be divided into two parts, a distribution transformer and an encoding function on patterns around the current position and a current input bit. Suppose that our input constraint is a 2D $(0,1)$ constraint(note that this is not a 2D RLL constraint according to our defintions here). Then it is very easy to see that for any 2D pattern there is a seqnece $\boldsymbol{x}$ such that the encoding rule of the bit stuffing together with $\boldsymbol{x}$ generates the 2D pattern. In this sense we can say that the bit stuffing algorithm is complete for the 2D $(0,1)$ constraint. We can also show that a bit stuffing algorithm for the 'nib' constraint, given in [12], is also complete, where the nib constraint requires that '0' is not surrounded by '1' and that '1' is not surrounded by '0.'

First we describe a general horizontal bit stuffing algorithm together with a local map. The horizontal bit stuffing algorithm is implemented by a function $f : \{+,-\}^{(2m+1)n+m} \times \{0,1\} \to \{+,-\}$, which we call an encoding function or a local map of the horizontal bit stuffing with scope $(2m+1)n+m$. We regard f as a map on a pattern of encoded bits at positions

$$\begin{aligned} D &= \{(i,j) : -m \le i \le m, -n \le j \le 1\} \\ &\quad \cup \{(i,0) : -m \le i \le -1\} \end{aligned}$$

and a current data bit. Suppose that $q(a,b)$, $0 \le a$, $0 \le b$ is a 2D pattern generated by the bit stuffing algorithm together with an input sequence $\boldsymbol{x} = x_0 x_1 \cdots$. For each position (a,b) the following is satisfied

$$q(a,b) = f(\mathcal{P}(a,b), x_\ell)$$

where $\mathcal{P}(a,b)$ is a pattern on positions

$$\{(a+i, b+j) : (i,j) \in D\}$$

and x_ℓ is a data bit used in determining $q(a,b)$ in the bit stuffing algorithm. Fig. 2 is an graphical explanation of the local map.

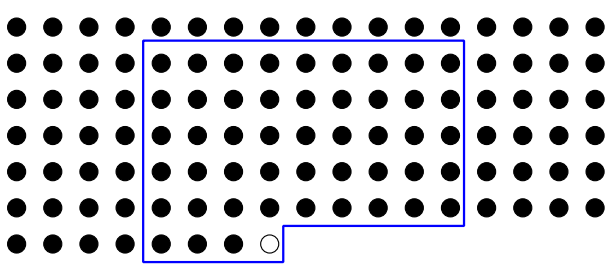

Fig. 2. Local map f of horizontal bit stuffing

When we are generating the pattern, we let $\ell \leftarrow \ell + 1$ if $f(\mathcal{P}(a,b),1) \neq f(\mathcal{P}(a,b),0)$ and otherwise we do not change ℓ. After determining an encoded bit at position (a,b) we move to the position $(a+1,b)$ and calculate

$$q(a+1,b) \leftarrow f(\mathcal{P}(a+1,b), x_\ell)$$

similarly. We continue this procedure to the rightmost position of a horizontal line. Precisely speaking we must describe how symbols on boundaries of a resulting pattern are defined because the local map has the finite scope. However we will omit these details in this paper.

The procedure defined above can be regarded as a function from a set of 1D data sequences to 2D patterns. The function can also be regarded as a variant of bit stuffing algorithm in [12], [13], [14]. Hence the procedure can be used in calculating lower bounds of the capacity of a given constrain [15]. However, we investigate the function as a representation of the constraint in this paper. The class of sofic systems includes all 1D constraints which appear in current practical applications[7]. Since a constraint in the class is an image of a constraint of finite type by a sliding block map and the constraint of finite type can be represented by a graph having a set of finite blocks satisfying the constraint as its vertex set, we might expect that the function defined above might play a role as a sliding block encoder(function) in 2D cases. We show that this is true in very restricted cases and is not true for many cases.

A. Completeness of horizontal bit stuffing

We prove the following theorem.

Theorem 2: Let k be a positive integer with $k \ge 2$. There is a local map f such that for every pattern satisfying a 2D $(1,k)$ RLL constraint, a horizontal bit stuffing encoder implemented by f with some input sequence can generate the pattern.

Proof: We describe an algorithm which can be implemented by some local map. Suppose that

- we are given a data sequence $x_1, x_2, \ldots$,
- data bits $x_1, x_2, \ldots, x_{\ell-1}$ are encoded already,
- an output symbol on the (i', j') position is denoted by $y(i', j')$,
- the encoder is calculating a symbol to be put on the (i,j) position.

We define $\overline{\alpha}$ by $\overline{\alpha} = |\alpha - 1|$ for $\alpha \in \{0,1\}$.

When determining symbols on the i-th row, there are positions on which symbols are uniquely determined from the vertical k constraint. We assume that symbols on these positions are determined before determining symbols by our function.

We apply the following steps sequentially to each position:

(1) if $y(i,j)$ is determined from the vertical k constraint, that is, there is a vertical run of symbol α of length k over the current position, then we put $y(i,j) \leftarrow \overline{\alpha}$, $i \leftarrow i+1$ and go to step (1);

(2) if the horizontal k constraint is violated by putting $y(i,j) = \alpha$ for some $\alpha \in \{0,1\}$, then we put $y(i,j) \leftarrow \overline{\alpha}$, $i \leftarrow i+1$ and go to step (1);

(3) we put $y(i,j) \leftarrow x_\ell$, $\ell \leftarrow \ell+1$, $i \leftarrow i+1$ and go to step (1).

By an induction on column and index i, we prove that we can always process one of the above steps. We assume that there is no violation of the given 2D $(1,k)$ RLL constraint.

Suppose that step (1) can not be processed properly, that is, there is a run of symbol α of length k over the (i,j) position but the horizontal constraint is violated by putting $y(i,j) = \overline{\alpha}$. There are two cases: (A) symbols $y(i,j-k_1), y(i,j-k_1+1), \ldots, y(i,j+k_2)$ are all determined from the vertical k constraint for some nonnegative integers k_1 and k_2 with $k_1 + k_2 = k$; (B) otherwise, that is, there is a k such that $y(i,j-k)$ is determined by step (2) or (3) with $1 \le k \le k_1$. If (A) is true then there should be a rectangle are consisting of only α of size $k \times (k+1)$. But this contradicts the assumption that $1, 2, \ldots, i-1$ columns are well determined. Next we consider the case (B). Let $\overline{k}$ be the smallest integer such that $1 \le \overline{k} \le k_1$ and $y(i,j-\overline{k})$ is determined by step (2) or (3). Since symbols $y(i,j-\overline{k}+1), \ldots, y(i,j+k_2)$ are uniquely determined from the vertical constraint, we could see that sequence $y(i,j-k_1)\cdots y(i,j+k_2)$ violates the horizontal k constraint when determining $y(i,j-\overline{k})$ and $y(i,j-\overline{k})$ should be determined by step (2). But if $y(i,j-\overline{k})$ could be determined by step (2) we note that $y(i,j-\overline{k})$ should be $\overline{\alpha}$. This contradicts that $y(i,j-k_2)\cdots y(i,j+k_2)$ is the run of symbol α. Therefore the case (B) can not happen.

Next we consider step (2). Suppose that step (2) does not work well, that is, the vertical constraint does not specify the symbol at the current position and both symbols, 0 and 1, violate the horizontal k constraint. This should mean $y(i,j-k)\cdots y(i,j-1)$ and $y(i,j+1)\cdots y(i,j+k)$ are runs of length k consisting of symbols α and $\overline{\alpha}$, respectively, for some $\alpha \in \{0,1\}$. This also means there is a block of size $k \times k$ consisting of only symbol $\overline{\alpha}$ over positions $(i,j-k), \ldots, (i,j-1)$. There is also a block of size $k \times k$ consisting of only symbol α over positions $(i,j+1), \ldots, (i,j+k)$. Then we could put no symbol at position $(i-1,j)$ because both symbols violate the horizontal constraint there. But this contradicts the induction hypothesis. Therefore we conclude that step (2) is well defined. It is obvious that step (3) is well defined.

Since if the 2D constraint does not specify a symbol then we can put data symbol randomly, it is clear that the above algorithm generate all possible 2D patterns satisfying the 2D constraint.

We can see that the above algorithm has finite scope, at most $k \times (2k+1)$.

We consider a decoder for this encoding procedure. When we look a symbol at the (i,j) position we can find a step by which the symbol is determined. Therefore we can decode the input data sequence correctly. ■

B. Incompleteness of horizontal bit stuffing

The following theorem says that the bit stuffing algorithm defined above is not enough to generate all possible patterns for all 2D (d,k) RLL constraints.

$$\begin{array}{cccccccccc}
- & - & + & + & - & - & + & + & - & - \\
+ & + & - & - & + & + & - & - & + & + \\
+ & + & - & - & + & + & - & - & + & + \\
+ & - & - & + & + & - & - & + & + & - \\
X & \ominus & & \oplus & & \ominus & & \oplus & & \ominus
\end{array}$$

Fig. 3. Impossible pattern for 2D $(2,4)$ RLL constraint (1)

Theorem 3: Let d and k be positive integers with $2 \le d$ and $2d \le k$. Let f be an encoding function of a horizontal bit stuffing algorithm for a 2D (d,k) RLL constraint of finite scope. Then there is a pattern which satisfies the 2D (d,k) RLL constraint but which can not be generated by f together with any input sequence $\boldsymbol{x}$.

Proof: We show that this theorem is true for a 2D $(2,4)$ RLL constraint. We can extend the following argument to all cases specified in the theorem.

Consider a pattern given in Fig. 3. Symbols with circles mean that the symbols are determined uniquely from the vertical $(2,4)$ RLL constraint. Suppose that we must determine a symbol at a position with symbol X. If we put symbol '+' there, we must have a pattern in Fig. 4 because the resulting pattern must obey the horizontal $(2,4)$ RLL constraint.

$$\begin{array}{cccccccccc}
- & - & + & + & - & - & + & + & - & - \\
+ & + & - & - & + & + & - & - & + & + \\
+ & + & - & - & + & + & - & - & + & + \\
+ & - & - & + & + & - & - & + & + & - \\
+ & \ominus & - & \oplus & + & \ominus & - & \oplus & + & \ominus
\end{array}$$

Fig. 4. Impossible pattern for 2D $(2,4)$ RLL constraint (2)

$$\begin{array}{cccccccccc}
- & - & + & + & - & - & + & + & - & - \\
+ & + & - & - & + & + & - & - & + & + \\
+ & + & - & - & + & + & - & - & + & + \\
+ & - & - & + & + & - & - & + & + & - \\
- & \ominus & + & \oplus & - & \ominus & + & \oplus & - & \ominus
\end{array}$$

Fig. 5. Impossible pattern for 2D $(2,4)$ RLL constraint (3)

If we put symbol '$-$' there, we can have a pattern in Fig. 5 because of the horizontal $(2,4)$ RLL constraint. But if the pattern in Fig. 3 is the partial pattern of a pattern given in Fig. 6 (we can obtain the pattern in Fig. 4 by dropping the rightmost column of the pattern in Fig. 6), then we should put '$-$' at the position with the symbol X. Notice that the length

$$\begin{array}{ccccccccccc}
- & - & + & + & - & - & + & + & - & - & - \\
+ & + & - & - & + & + & - & - & + & + & - \\
+ & + & - & - & + & + & - & - & + & + & - \\
+ & - & - & + & + & - & - & + & + & - & - \\
- & \ominus & + & \oplus & - & \ominus & + & \oplus & - & \ominus & \oplus
\end{array}$$

Fig. 6. Impossible pattern for 2D $(2,4)$ RLL constraint (4)

of the pattern in Fig. 3 can be arbitrarily long.

Suppose that the horizontal length of the pattern in Fig. 3 is larger than the width of the scope of f. Then a problem arises: which symbol should we put at the position of X? If we always put '$-$' there then the constraint will never be violated but we can not generate the following pattern

$$\begin{array}{ccccccccccc} - & - & + & + & - & - & + & + & - & - & + \\ + & + & - & - & + & + & - & - & + & + & + \\ + & + & - & - & + & + & - & - & + & + & + \\ + & - & - & + & + & - & - & + & + & - & - \\ + & \ominus & - & \oplus & + & \ominus & - & \oplus & + & \ominus & - \end{array}.$$

If we put '$+$' there without looking the rightmost column then we violate the input constraint when we have a pattern similar to the pattern in Fig. 5.

This shows that the width of the scope of f can not be fixed in advance. That is, there is no local map by which we can generate all constrained pattern essentially.

To complete our proof of the theorem, we must show that for every pattern appeared above there is an infinite 2D pattern which contains the pattern as a partial pattern. Although we consider only pattern in Fig. 4 here, we can give the same proofs for other patterns appeared in the above discussion.

First we add two same lines on the top of the pattern. They are obtained by reversing each symbol. Similarly we add two lines at the bottom of the patterns. They are obtained by reversing the last line symbol by symbol.

$$\begin{array}{ccccccccccc} - & - & + & + & - & - & + & + & - & - & - \\ - & - & + & + & - & - & + & + & - & - & - \\ + & + & - & - & + & + & - & - & + & + & + \\ + & + & - & - & + & + & - & - & + & + & + \\ + & - & - & + & + & - & - & + & + & - & - \\ + & \ominus & - & \oplus & + & \ominus & - & \oplus & + & \ominus & - \\ - & + & + & - & - & + & + & - & - & + & + \\ - & + & + & - & - & + & + & - & - & + & + \end{array}$$

Next we add two vertical lines to the right hand side of the pattern. The two lines are obtained by reversing the rightmost line of the pattern. Then we also add two vertical lines to

$$\begin{array}{ccccccccccccc} - & - & + & + & - & - & + & + & - & - & - & + & + \\ - & - & + & + & - & - & + & + & - & - & - & + & + \\ + & + & - & - & + & + & - & - & + & + & + & - & - \\ + & + & - & - & + & + & - & - & + & + & + & - & - \\ + & - & - & + & + & - & - & + & + & - & - & + & + \\ + & \ominus & - & \oplus & + & \ominus & - & \oplus & + & \ominus & - & + & + \\ - & + & + & - & - & + & + & - & - & + & + & - & - \\ - & + & + & - & - & + & + & - & - & + & + & - & - \end{array}$$

the left hand side of the pattern. The two lines are obtained by reversing symbols in the leftmost line of the pattern(see Fig. 7).

Next we flip the leftmost and the second leftmost vertical lines infinitely many times. We also flip the rightmost and the second rightmost vertical lines infinitely many times. We can extend the pattern along the vertical direction by repeating horizontal lines(see Fig. 8) and, then, we can define an infinite pattern which satisfies a 2D $(2,4)$ RLL constraint and contains a pattern given in Fig. 3.

■

The following theorem says that if we use a restricted local map for our bit stuffing algorithm, then the algorithm is incomplete even for a 2D $(1,k)$ RLL constraint.

Theorem 4: Let k, m and n be positive integers and assume that $k \geq 2$. Let f be any local map defined on positions

$$\begin{aligned} D' &= \{(i,j) : -m \leq i \leq 0, -n \leq j \leq 1\} \\ &\quad \cup \{(i,0) : -m \leq i \leq -1\}. \end{aligned}$$

Then a horizontal bit stuffing encoder implemented by f is incomplete for a 2D $(1,k)$ RLL constraint, that is, there is a pattern which can not be generated by f with any data sequence $\boldsymbol{x}$.

Proof: We assume that $k = 2$. We consider the following pattern P1. We can see that P1 is a part of an infinite 2D pattern satisfying a 2D $(1,2)$ RLL constraint by concatenating flipped rows and columns in P1.

$$\begin{array}{cccccccc} - & + & - & - & + & - & + & - \\ + & - & + & + & - & + & - & + \\ - & + & - & + & + & - & + & - \\ + & - & \oplus & - & + & - & + & - \end{array}$$

A position of a circled '$+$' symbol is denoted by (a,b). We define $\mathcal{P}_1$ to be a finite sub-pattern on a set of positions

$$\begin{aligned} B_1 &= \{(i+a, i+b) : -m \leq i \leq 0, -n \leq j \leq 1\} \\ &\quad \cup \{(i+a, b) : -m \leq i \leq -1\}. \end{aligned}$$

Then we can consider f maps $\mathcal{P}_1$ and a current data symbol to symbol '$+$.'

Next we consider the following pattern P2

$$\begin{array}{cccccccc} - & + & - & - & + & - & + & - \\ + & - & + & + & - & + & - & + \\ - & + & - & + & + & - & + & - \\ + & - & \ominus & X & & & & \end{array}.$$

We can put no symbol at a position with X because '$+$' and '$-$' symbols put there violate the vertical and horizontal $(1,2)$ RLL constraints, respectively. Note that a pattern in P2 on positions B_1 is the same as $\mathcal{P}_1$. This means that f maps $\mathcal{P}_1$ to '$+$' symbol independently of the current input symbol.

We also have the following infinite pattern P3 satisfying the 2D $(1,2)$ RLL constraint

$$\begin{array}{ccccccc} - & + & - & + & - & + & - \\ + & - & + & - & + & - & + \\ - & + & - & + & - & + & - \\ + & - & \ominus & + & - & - & + \\ - & + & + & - & + & + & - \end{array}.$$

We note that the finite partial pattern in P3 on positions B_1 is the same as $\mathcal{P}_1$. Therefore we conclude that P3 can not be produced by our algorithm with f.

We can extend the above argument to 2D $(1, k)$ RLL constraints for any $k \geq 2$. For example, if $k = 5$ we have the following pattern as P1.

$$\begin{array}{cccccccccc}
+ & - & + & - & + & - & - & + & - & + \\
- & + & - & + & - & + & + & - & + & - \\
+ & - & + & - & + & - & + & + & - & + \\
- & + & - & + & - & + & + & - & + & - \\
+ & - & + & - & + & - & + & + & - & + \\
- & + & - & + & - & + & + & - & + & - \\
+ & - & - & - & - & \oplus & - & + & - & + \\
- & + & + & + & + & - & + & - & + & -
\end{array}$$

■

VI. Conclusion

We have proposed encoding algorithms for two dimensional weak input constraints. We have given a formal definition of a horizontal bit stuffing algorithm which can be considered as a generalization the proposed encoding algorithm, e.g., Algorithm 1. The formal definition can be regarded as a representation of a 2D RLL constraint as well as an encoding algorithm for the 2D RLL constraint. We have shown that the horizontal bit stuffing is complete for 2D $(1, k)$ RLL constraints but incomplete for 2D (d, k) RLL constraints if $2 \leq d$ and $2d \leq k$. We have also shown that the algorithm is incomplete even for 2D $(1, k)$ RLL constraints if we use a restricted local map as given in Theorem 4.

Acknowledgment

This work was supported in part by SRC(Storage Research Consortium). The author also would like to thank Professor T. Hamachi for pointing out that the the Algorithm 1 can be regarded as a representation of a 2D RLL constraint.

References

[1] R. Talyansky, T. Etzion, and R. Roth, "Efficient code constructions for certain two-dimensional constraints," *IEEE Trans. Inform. Theory*, vol. IT-45, pp. 794–799, 1999.

[2] A. Kato and K. Zeger, "On the Capacity of Two-Dimensional Run-Length Constrained Channels," *IEEE Trans. Inform. Theory*, vol. IT-45, pp. 1527–1540, 1999.

[3] T. Ichihashi and H. Kamabe, "The asymptotic property of the rate of two dimensional balanced code," *Technical Report of IEICE*, vol. IT99-235, pp. 43–47, July 1999.

[4] T. Etzion and K. G. Paterson, "Zero/Positive Capacities of Two-Dimensional Runlength-Constrained Arrays," *IEEE Trans. Inform. Theory*, pp. 3186–3199, Sep 2005.

[5] W. Weeks and R. E. Blahut, "The Capacity and Coding Gain of Certain Checkerboard Codes," *IEEE Trans. Inform. Theory*, vol. IT-44, pp. 1193–1203, 1998.

[6] R. Adler, D. Coppersmith and M. Hassner, "Algorithms for sliding block codes," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 5–22, January 1983.

[7] B. Marcus and D. Lind, *Symbolic dynamics and coding*, Cambridge, 1995.

[8] Min Jin, K. A. S. Immink, B. Farhang-Boroujeny, "Design Techniques for Weakly Constrained Codes," *IEEE Trans. Commun.*, vol. 51, pp. 709–714, 2003.

[9] K. A. S. Immink, *Codes for Mass Data Storage Systems, 2nd ed.*, Shannon Foundation Publishers, 2004.

[10] H. Kamabe, "Encoding Algorithms for 2 Dimensional Run-Length-Limited Constraints," *Proceedings of 2004 IEEE ISIT*, p. 193, 2004.

[11] K. A. S. Immink, "Weakly constrained codes," *Electron. Lett.*, vol. 33, pp. 1943–1944, 1997.

[12] R. Roth, P. Siegel, and J. Wolf, "Efficient coding schemes for hard-square model," *IEEE Trans. Inform. Theory*, vol. IT-47, pp. 1166–1176, 2001.

[13] S. Halevy, J. Chen, R. Roth, P. Siegel, and J. Wolf, "Improved Bit-Stuffing Bounds on Two-Dimensional Constraints," *IEEE Trans. Inform. Theory*, vol. IT-50, pp. 824–838, 2004.

[14] S. Aviran, P. Siegel, and J. Wolf, "Two-Dimensional Bit-Stuffing Schemes with Multiple Transformers," *Proceedings of ISIT 2005*, pp. 1478–1482, Sep 2005.

[15] H. Kamabe, "Encoding Algorithms for 2 Dimensional Weak Run-Length-Limited Constraints ," *the Proceedings of ITW 2006*, pp. 375–379, Chengdu, 2006.

$$
\begin{array}{cccccccccccccccc}
+ & + & - & - & - & + & + & - & - & + & + & - & - & - & + & + \\
+ & + & - & - & - & + & + & - & - & + & + & - & - & - & + & + \\
- & - & + & + & + & - & - & + & + & - & - & + & + & + & - & - \\
- & - & + & + & + & - & - & + & + & - & - & + & + & + & - & - \\
- & - & + & + & - & - & + & + & - & - & + & + & - & - & + & + \\
- & - & + & + & \ominus & - & \oplus & + & \ominus & - & \oplus & + & \ominus & - & + & + \\
+ & + & - & - & + & + & - & - & + & + & - & - & + & + & - & - \\
+ & + & - & - & + & + & - & - & + & + & - & - & + & + & - & -
\end{array}
$$

Fig. 7.

$$
\begin{array}{ccccccccccccccccccc}
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\cdots & - & - & + & + & + & - & - & + & + & - & - & + & + & + & - & - & + & \cdots \\
\cdots & - & - & + & + & + & - & - & + & + & - & - & + & + & + & - & - & + & \cdots \\
\cdots & + & + & - & - & - & + & + & - & - & + & + & - & - & - & + & + & - & \cdots \\
\cdots & + & + & - & - & - & + & + & - & - & + & + & - & - & - & + & + & - & \cdots \\
\cdots & - & - & + & + & + & - & - & + & + & - & - & + & + & + & - & - & + & \cdots \\
\cdots & - & - & + & + & + & - & - & + & + & - & - & + & + & + & - & - & + & \cdots \\
\cdots & - & - & + & + & - & - & + & + & - & - & + & + & - & - & + & + & - & \cdots \\
\cdots & - & - & + & + & \ominus & - & \oplus & + & \ominus & - & \oplus & + & \ominus & - & + & + & - & \cdots \\
\cdots & + & + & - & - & + & + & - & - & + & + & - & - & + & + & - & - & + & \cdots \\
\cdots & + & + & - & - & + & + & - & - & + & + & - & - & + & + & - & - & + & \cdots \\
\cdots & - & - & + & + & - & - & + & + & - & - & + & + & - & - & + & + & - & \cdots \\
\cdots & - & - & + & + & - & - & + & + & - & - & + & + & - & - & + & + & - & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}
$$

Fig. 8. Flipping horizontal lines infinitely many times