On q-ary Antipodal Matchings and Applications

Erik Ordentlich
Hewlett–Packard Laboratories
Palo Alto, CA 94304
erik.ordentlich@hp.com

Ron M. Roth
Computer Science Department
Technion, Haifa, Israel
ronny@cs.technion.ac.il

Gadiel Seroussi
Hewlett–Packard Laboratories
Palo Alto, CA 94304
gadiel.seroussi@hp.com

***Abstract*—We define a q-ary antipodal matching to be a perfect matching in the bipartite graph with vertices corresponding to words of length ℓ over the integer alphabet $\mathcal{Q} = \{0, 1, \ldots, q-1\}$, wherein the left and right vertices are those with respective component sums greater and smaller than $\ell(q-1)/2$, and wherein two vertices are connected by an edge if one of the corresponding words dominates the other. We present two different constructions of efficiently computable q-ary antipodal matchings. We then show how such matchings can be used for encoding arbitrary data into $n \times n$ arrays over the alphabet $\mathcal{Q}$ all of whose row and column sums are at most $n(q-1)/2$. Such encoders might be useful for mitigating parasitic currents in a next generation memory technology based on crossbar arrays of resistive devices.**

I. Introduction

For any positive real r, let $[r]$ denote the integer set $\{i \in \mathbb{Z} : 0 \le i < r\}$. Hereafter, we fix $\mathcal{Q}$ to be the integer alphabet $[q]$, for some integer $q \ge 2$. For a symbol $x \in \mathcal{Q}$, we define its *cost* by $\mathsf{c}(x) = 2x - (q-1)$ (which may be negative) and its *complement* (over $\mathcal{Q}$) by $\overline{x} = q-1-x$; thus, $\mathsf{c}(\overline{x}) = -\mathsf{c}(x)$. We extend these definitions to words $\boldsymbol{x} = (x_i)_{i\in[\ell]}$ over $\mathcal{Q}$ by defining $\mathsf{c}(\boldsymbol{x}) = \sum_{i\in[\ell]} \mathsf{c}(x_i) = 2\big(\sum_{i\in[\ell]} x_i\big) - \ell(q-1)$ and $\overline{\boldsymbol{x}} = (\overline{x}_i)_{i\in[\ell]}$.

Consider the *bipartite graph* (cf. [3]) $\mathcal{G}_{\ell,q} = (\mathcal{L} : \mathcal{R}, \mathcal{E})$ with left and right vertex sets $\mathcal{L}$ and $\mathcal{R}$ and edges $\mathcal{E}$ given by

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{\ell,q} = \big\{\boldsymbol{x} \in \mathcal{Q}^\ell \,:\, \mathsf{c}(\boldsymbol{x}) > 0\big\} \\ \mathcal{R} &= \mathcal{R}_{\ell,q} = \big\{\boldsymbol{x} \in \mathcal{Q}^\ell \,:\, \mathsf{c}(\boldsymbol{x}) < 0\big\} \\ \mathcal{E} &= \mathcal{E}_{\ell,q} = \big\{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{L} \times \mathcal{R} \,:\, \boldsymbol{x} \ge \boldsymbol{y}\big\} \ , \end{aligned}$$

where the last inequality holds componentwise. We define a *q-ary antipodal matching* to be a perfect matching in $\mathcal{G}_{\ell,q}$, wherein a perfect matching is the usual notion from graph theory, namely, an edge subset $\{(\boldsymbol{x}, \varphi(\boldsymbol{x})) : \boldsymbol{x} \in \mathcal{L}\}$, where $\varphi : \mathcal{L} \to \mathcal{R}$ is a bijection.

Antipodal matchings for the special case of $q = 2$ were studied recently in [10]. As noted therein, for this case, the constituent bipartite graphs obtained by decomposing the graph $\mathcal{G}_{\ell,q}$ according to the cost of left and right vertices, are regular graphs, and hence it follows from Hall's theorem [3, pp. 217–218] that perfect matchings exist. An explicit construction of a perfect matching for the case of $q = 2$ was also presented. In the case of $q > 2$, there does not seem to be an analogous regular decomposition of $\mathcal{G}_{\ell,q}$, or at least one that we are aware of, and thus, in the general case, it is not immediately evident that a perfect matching even exists.

It turns out that perfect matchings do in fact exist for all q and our first set of results pertains to the constructions of two efficiently computable q-ary antipodal matchings, by which we mean that the bijection φ and its inverse are efficiently computable. For reasons related to the specifics of the constructions, we shall refer to the two constructions, respectively, as the *unary* and *complementary* (q-ary antipodal) matchings. The complementary matching will satisfy, in addition, the following *complementary property*: if an edge $(\boldsymbol{x}, \boldsymbol{y})$ belongs to the matching, then $y_i \in \{x_i, \overline{x}_i\}$ for each $i \in [\ell]$. Though we lack the space to explain this here, complementary antipodal matchings exist in more general bipartite graphs than $\mathcal{G}_{\ell,q}$ and these graphs may have some potential advantages in the context of our main application for these matchings, which, as explained further below, is the construction of codes that might be useful for mitigating parasitic currents in a next generation memory technology. On the negative side, although we will show that matchings satisfying the complementary property exist for all q, we presently have explicit descriptions of such matchings only for limited q (still, when q is *fixed*, the matching can be computed in time complexity that is linear in ℓ). Unary matchings, on the other hand, are specific to the graph $\mathcal{G}_{\ell,q}$, but can be explicitly specified for all q, and can be computed requiring only $O(\ell)$ arithmetic operations on integers with absolute value at most $\ell(q-1)$.

Consider the set $\mathcal{A}_{n\times n}$ of all $n \times n$ arrays $A = (A_{i,j})_{i,j\in[n]}$ over $\mathcal{Q}$ such that the cost of each row and column is nonpositive, namely, for every $i, j \in [n]$,

$$\mathsf{c}\left(A_{i,[n]}\right) \le 0 \quad \text{and} \quad \mathsf{c}\left(A_{[n],j}\right) \le 0 \ , \tag{1}$$

where the notation $A_{i,[\ell]}$ (respectively, $A_{[\ell],j}$) stands for the row (respectively, column) word formed by the first ℓ entries of row i (respectively, column j) of an array A. Our principal application of q-ary antipodal matchings is to the problem of efficiently encoding and decoding arbitrary data to and from (a subset of) $\mathcal{A}_{n\times n}$. Following the usual formal definitions, a code for this problem (referred to as a *row–column cost limiting code*) consists of a subset $\mathbb{C} \subseteq \mathcal{A}_{n\times n}$, an encoder (bijection) from $[|\mathbb{C}|]$ to $\mathbb{C}$, and a respective decoder (inverse mapping). Of interest are codes for which the encoder and decoder can be computed with low complexity and $\mathbb{C}$ is as large as possible. Obviously, $|\mathbb{C}| \le |\mathcal{A}_{n\times n}| < q^{n^2}$, and we shall refer to the gap $n^2 \log_2 q - \log_2 |\mathbb{C}|$ as the *redundancy*

The work of R. M. Roth was carried out in part while visiting Hewlett–Packard Laboratories, Palo Alto, CA. This work was supported in part by Grant No. 2008058 from the United-States–Israel Binational Science Foundation.

of the code.

Our second set of results addresses this coding problem, by generalizing the row–column cost limiting codes of [10] to the q-ary case, for $q > 2$. For the q-ary iterative flipping code (generalizing the iterative flipping code from [10]), the upper bound on the number of row–column flips during encoding increases by a factor of $q-1$. This provides even stronger motivation for the q-ary antipodal matching codes based on the above q-ary antipodal matchings, which avoid iterations. Both the q-ary iterative flipping and q-ary antipodal matching codes constructed continue to have a redundancy of roughly $2n$ bits, as in the binary case. Thus, one new aspect of the $q > 2$ setting, as will be seen, is that the q-ary array positions that carry the redundancy also carry non-redundant information.

The motivation for encoding data into the constrained arrays $\mathcal{A}_{n\times n}$ is, as in [10], to reduce parasitic current in a next generation memory technology based on crossbar arrays of resistive memory devices [9], [11]. The codes presented here extend this paradigm to devices which can be programmed into $q > 2$ resistance states. The rough idea is as follows. In the q-ary setting, the states of the devices in an $n\times n$ crossbar can be represented by an $n\times n$ array over $\mathcal{Q}$. For a certain writing biasing scheme and overwriting protocol, the parasitic current magnitude passing through any device can be expressed as a non-decreasing function $\varrho : \mathcal{Q} \to \mathbb{R}^+$ of the device state, and the maximum parasitic current magnitude passing through any wire of the crossbar can be bounded by the maximum sum of $\varrho(A_{i,j})$ over all rows and columns of the array. The parasitic current is thus data-dependent and, by encoding data using the above codes to limit row and column costs of the resulting arrays, the worst case current can be reduced. In the case that ϱ is an affine function of the state, the constraint (1) is well matched to ϱ. As alluded to, in the general case, codes based on complementary matchings adapted to a different ϱ-dependent bipartite graph may have better current limiting properties than the above. For conciseness, here we will target only the above simpler scenario with affine ϱ.

II. q-ARY ANTIPODAL MATCHINGS

Both of our q-ary antipodal matching constructions are based on reductions to the binary case and, at least conceptually, invoke the binary antipodal matching of [10] as a subroutine. The constructions differ in the nature of the reduction.

A. The binary case

Let $\mathcal{B}$ denote the binary (integer) alphabet $\{0, 1\}$. We include here a brief review of the binary antipodal matching from [10], whose bijection we shall denote by φ_{B}. In this subsection, the definitions of cost and complementation correspond to $q = 2$, i.e., $\mathsf{c}(0) = -1$, $\mathsf{c}(1) = 1$, $\overline{0} = 1$, and $\overline{1} = 0$.

Given a word $\boldsymbol{a} = (a_i)_{i\in[m]}$ in $\mathcal{B}^m$ with $\mathsf{c}(\boldsymbol{a}) > 0$, the image $\varphi_{\mathsf{B}}(\boldsymbol{a})$ is defined as follows. A position $i \in [m]$ in $\boldsymbol{a}$ is called *minimal* (with respect to $\boldsymbol{a}$) if and only if $\sum_{j=i}^{i+h} \mathsf{c}(a_j) > 0$ for every natural h, where indices are wrapped around modulo m. Let $\mathcal{P}_{\boldsymbol{a}}$ denote the set of minimal positions. Note that $\mathcal{P}_{\boldsymbol{a}}$ is nonempty whenever $\mathsf{c}(\boldsymbol{a}) > 0$ and that $a_i = 1$ for every $i \in \mathcal{P}_{\boldsymbol{a}}$. For $\boldsymbol{a} \in \mathcal{B}^m$ with $\mathsf{c}(\boldsymbol{a}) > 0$, the mapping $\boldsymbol{a} \mapsto \varphi_{\mathsf{B}}(\boldsymbol{a})$ is then defined by

$$(\varphi_{\mathsf{B}}(\boldsymbol{a}))_i = \begin{cases} a_i & \text{if } i \notin \mathcal{P}_{\boldsymbol{a}} \\ \overline{a}_i & \text{if } i \in \mathcal{P}_{\boldsymbol{a}} \end{cases}, \quad i \in [m] . \tag{2}$$

In order to compute this mapping, the set $\mathcal{P}_{\boldsymbol{a}}$ must be determined. There is an algorithm for finding the minimal positions that runs in time that is linear in ℓ and is described in Figure 4 of [10] (see also [5] and [6]). It follows from the results of [10] that φ_{B} is a bijection from $\{\boldsymbol{a} \in \mathcal{B}^m : \mathsf{c}(\boldsymbol{a}) > 0\}$ to $\{\boldsymbol{a} \in \mathcal{B}^m : \mathsf{c}(\boldsymbol{a}) < 0\}$ and, so, it induces a perfect matching in the graph $\mathcal{G}_{m,2}$.

B. Unary matching

The bijection underlying the unary matching will be denoted by φ_{U}. It is based on the following unary coding of $\mathcal{Q}$ into words in $\mathcal{B}^{q-1}$:

$$\mathsf{u}(x) = \underbrace{0\,0\cdots 0}_{\overline{x}}\underbrace{1\,1\cdots 1}_{x}, \quad x \in \mathcal{Q} .$$

Namely, a symbol $x \in \mathcal{Q}$ is mapped to a binary word of length $q-1$ with $\overline{x} = q-1-x$ initial 0's followed by x trailing 1's. It is obvious that u is invertible. The mapping u can be extended to words $\boldsymbol{x} = (x_i)_{i\in[\ell]}$ in $\mathcal{Q}^\ell$ via $\mathsf{u}(\boldsymbol{x}) = \mathsf{u}(x_0)\,\mathsf{u}(x_1)\,\ldots\,\mathsf{u}(x_{\ell-1})$ (the concatenation of the symbol-wise mappings). This word extension, $\mathsf{u} : \mathcal{Q}^\ell \to \mathsf{u}(\mathcal{Q}^\ell)$, is also readily seen to be invertible. The following lemma can be easily shown to follow from the properties of u and φ_{B}.

Lemma 2.1: For every $\boldsymbol{x} \in \mathcal{L}_{\ell,q}$ (the left vertex set in $\mathcal{G}_{\ell,q}$),

$$\varphi_{\mathsf{B}}(\mathsf{u}(\boldsymbol{x})) \in \mathsf{u}(\mathcal{R}_{\ell,q}) ,$$

where φ_{B} is the bijection (2) defined for binary words of length $m = \ell(q-1)$.

Lemma 2.1, in turn, allows us to define the following mapping $\varphi_{\mathsf{U}} : \mathcal{L}_{\ell,q} \to \mathcal{R}_{\ell,q}$:

$$\varphi_{\mathsf{U}}(\boldsymbol{x}) = \mathsf{u}^{-1}(\varphi_{\mathsf{B}}(\mathsf{u}(\boldsymbol{x}))) , \quad \boldsymbol{x} \in \mathcal{L}_{\ell,q} . \tag{3}$$

In the next proposition, $\boldsymbol{x}^*$ stands for the word obtained by reversing the order of entries in the word complement $\overline{\boldsymbol{x}}$ of $\boldsymbol{x}$.

Proposition 2.2: The mapping $\varphi_{\mathsf{U}} : \mathcal{L}_{\ell,q} \to \mathcal{R}_{\ell,q}$ is a bijection, and, so, $\{(\boldsymbol{x}, \varphi_{\mathsf{U}}(\boldsymbol{x})) : \boldsymbol{x} \in \mathcal{L}_{\ell,q}\}$ is a perfect matching in $\mathcal{G}_{\ell,q}$. Moreover, $\varphi_{\mathsf{U}}^{-1}(\boldsymbol{y}) = (\varphi_{\mathsf{U}}(\boldsymbol{y}^*))^*$ for every $\boldsymbol{y} \in \mathcal{R}_{\ell,q}$.

We point out that this construction can also be inferred from known explicit minimal partitions of $\mathcal{Q}^\ell$ into symmetric chains [2], [4], [7].

Computing φ_{U} via φ_{B} as in (3) has complexity proportional to $q\ell$ operations if the linear time algorithm from [10] is used for computing φ_{B}. By exploiting constraints on the binary words in the range of u and properties of φ_{B}, it is possible to carry out the computation more efficiently, with only a constant (independent of q) times ℓ operations on integers with absolute value at most $\ell(q-1)$. The algorithm is given in Figure 1. Roughly speaking, the first loop "identifies" an element of $\mathcal{P}_{\mathsf{u}(\boldsymbol{x})}$ and the next loop identifies the remaining elements and implements the bijection directly in the $\mathcal{Q}$-domain. The following proposition can be proved.

Proposition 2.3: The algorithm of Figure 1 computes φ_{U}.

```
Input: x = (x_i)_{i∈[ℓ]} ∈ L_{ℓ,q}
Output: y = φ_U(x)
Variables: i, t ∈ Z/ℓZ (arithmetic modulo ℓ);  d, S, W ∈ Z

t = 0;  d = 0;  S = 0;
for i = 1 to ℓ − 1 do
    S += (c(x_{i−1}) + c(x_i))/2;
    if S ≤ d then
        d = S;   t = i;
    end if
end for
y = x;   W = c(x);
while W > 0 do
    d = 0;
    while d ≤ 0 do
        d += (c(x_{t−1}) + c(x_t))/2;   t −−;
    end while
    y_t = x_t − d;   W −= d;
end while
return y.
```

Fig. 1. Efficient computation of $\varphi_{\mathsf{U}}(\boldsymbol{x})$.

C. Complementary matching: existence

For a word $\boldsymbol{z} = (z_i)_{i \in [\ell]}$ in $[q/2]^\ell$, denote by $\mathcal{M}_q(\boldsymbol{z})$ the subset $\{z_0, \overline{z}_0\} \times \{z_1, \overline{z}_1\} \times \ldots \times \{z_{\ell-1}, \overline{z}_{\ell-1}\}$ (of size 2^ℓ) of $\mathcal{Q}^\ell$, and define $\mathcal{L}_q(\boldsymbol{z}) = \mathcal{L}_{\ell,q} \cap \mathcal{M}_q(\boldsymbol{z})$ and $\mathcal{R}_q(\boldsymbol{z}) = \mathcal{R}_{\ell,q} \cap \mathcal{M}_q(\boldsymbol{z})$. Clearly, for $\boldsymbol{z}$ ranging over $[q/2]^\ell$, the sets $\mathcal{L}_q(\boldsymbol{z})$ (respectively, $\mathcal{R}_q(\boldsymbol{z})$) form a partition of $\mathcal{L}_{\ell,q}$ (respectively, $\mathcal{R}_{\ell,q}$). For any $\boldsymbol{z}$ as above, let $\mathcal{G}_q(\boldsymbol{z})$ be the bipartite subgraph of $\mathcal{G}_{\ell,q}$ that is induced by the vertices in $\mathcal{L}_q(\boldsymbol{z}) \cup \mathcal{R}_q(\boldsymbol{z})$. An example of a graph $\mathcal{G}_q(\boldsymbol{z})$ is shown in Figure 2.

By the complementary property of complementary matchings in $\mathcal{G}_{\ell,q}$ (defined in Section I), to show their existence, it suffices to show that a perfect matching exists in $\mathcal{G}_q(\boldsymbol{z})$ for all q, ℓ, and all $\boldsymbol{z} \in [q/2]^\ell$. This will follow from the next slightly more general matching result, which we prove below.

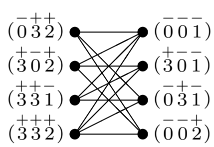

Fig. 2. Graph $\mathcal{G}_4(0\,0\,1)$.

Proposition 2.4: Given a vector $\boldsymbol{w} \in \mathbb{R}^m$ with nonnegative entries, let σ be the sum of entries of $\boldsymbol{w}$ and let the bipartite graph $G = G(\boldsymbol{w}) = (L : R, E)$ be defined by

$$\begin{array}{lllll} L & = & L(\boldsymbol{w}) & = & \{\boldsymbol{a} \in \mathcal{B}^m \,:\, \langle \boldsymbol{a}, \boldsymbol{w} \rangle > \sigma/2\} \\ R & = & R(\boldsymbol{w}) & = & \{\boldsymbol{a} \in \mathcal{B}^m \,:\, \langle \boldsymbol{a}, \boldsymbol{w} \rangle < \sigma/2\} \\ E & = & E(\boldsymbol{w}) & = & \{(\boldsymbol{a}, \boldsymbol{b}) \in L \times R \,:\, \boldsymbol{a} \geq \boldsymbol{b}\} \end{array} \quad ,$$

where $\langle \cdot, \cdot \rangle$ denotes inner product. Then G exhibits a perfect matching.

The existence of the desired perfect matching in $\mathcal{G}_q(\boldsymbol{z})$ then follows since this graph is isomorphic to the graph $G(\boldsymbol{w})$ of Proposition 2.4 with $\boldsymbol{w} = -\big(\mathsf{c}(z_i)\big)_{i \in [\ell]}$. The graph $G(\boldsymbol{w})$ corresponding to the example $\mathcal{G}_q(\boldsymbol{z})$ of Figure 2 is also shown in the figure, where, for better readability, the binary values $0, 1$ are represented, respectively, as $-, +$.

Observe that for the case where all the entries of $\boldsymbol{w}$ are equal, Proposition 2.4 follows from the binary antipodal matching results. In the general case, we prove Proposition 2.4 by verifying the condition for the existence of a perfect matching in Hall's theorem, which is that all subsets of left vertices have at least as many right neighbors. Unlike the case where all entries of $\boldsymbol{w}$ are equal, generally, we are not able to partition the graph $G(\boldsymbol{w})$ into a collection of *regular* induced subgraphs and, thus, checking Hall's condition is somewhat more involved. Key to our proof is the following classical combinatorial lemma concerning set differences. Given two sets α and β, denote by $\alpha \setminus \beta$ their set difference $\{s \in \alpha \,:\, s \notin \beta\}$. For a finite collection of sets X, let $\Delta X = \{\alpha \backslash \beta \,:\, \alpha, \beta \in X\}$ denote the collection of distinct differences of members of X.

Lemma 2.5 (Marica–Schönheim inequality [8]):

$$|\Delta X| \geq |X| \; .$$

Proof of Proposition 2.4: We verify that $G(\boldsymbol{w})$ satisfies Hall's condition. Let $X \subseteq L(\boldsymbol{w})$ and let $\mathcal{N}(X) \subseteq R(\boldsymbol{w})$ denote the set of neighbors of X. Consider the set

$$Y = \big\{ \boldsymbol{a} \wedge \overline{\boldsymbol{b}} \,:\, \boldsymbol{a}, \boldsymbol{b} \in X \big\} \; , \tag{4}$$

where $\overline{\boldsymbol{b}}$ denotes the (Boolean) word complement of $\boldsymbol{b}$ over $\mathcal{B}$ and "$\wedge$" denotes componentwise conjunction ("AND"). By regarding the words of $L(\boldsymbol{w})$ as characteristic vectors of subsets of $[m]$, it follows from Lemma 2.5 that $|Y| \geq |X|$. We next argue that $Y \subseteq \mathcal{N}(X)$. To see this, consider a word $\boldsymbol{v}$ in Y. By (4), we can write $\boldsymbol{v} = \boldsymbol{a} \wedge \overline{\boldsymbol{b}}$ for some $\boldsymbol{a}, \boldsymbol{b} \in X$, implying that $\boldsymbol{v} \leq \boldsymbol{a}$ and $\overline{\boldsymbol{v}} \geq \boldsymbol{b}$. From the latter inequality, we have $\langle \boldsymbol{w}, \overline{\boldsymbol{v}} \rangle \geq \langle \boldsymbol{w}, \boldsymbol{b} \rangle > \sigma/2$ so that $\langle \boldsymbol{w}, \boldsymbol{v} \rangle < \sigma/2$, and this, together with $\boldsymbol{v} \leq \boldsymbol{a}$, implies that $\boldsymbol{v} \in \mathcal{N}(X)$ (since $\boldsymbol{a} \in X$) or that $Y \subseteq \mathcal{N}(X)$. Putting everything together, we have $|\mathcal{N}(X)| \geq |Y| \geq |X|$, thus verifying Hall's condition and completing the proof. ■

Using essentially the above technique (but with a novel variation of Lemma 2.5) we can also prove the following more general matching result, which may be of independent interest.

Proposition 2.6: Let $\mathsf{G} = (\mathsf{L} : \mathsf{R}, \mathsf{E})$ be a bipartite graph where L and R form any partition of $\mathcal{B}^m$ with the property that $\boldsymbol{a} \in \mathsf{L}$ if and only if $\overline{\boldsymbol{a}} \in \mathsf{R}$, and where the edge set is

$$\mathsf{E} = \{(\boldsymbol{a}, \boldsymbol{b}) \in \mathsf{L} \times \mathsf{R} \,:\, \boldsymbol{a} \leq \boldsymbol{b} \text{ or } \boldsymbol{b} \leq \boldsymbol{a}\} \; .$$

Then G exhibits a perfect matching.

D. Complementary matching: construction

The existence result derived from Proposition 2.4 does not provide, in general, an efficient way for constructing the claimed perfect matching for $\mathcal{G}_q(\boldsymbol{z})$ for arbitrary values of q and ℓ. In this subsection, we apply Proposition 2.4 in a slightly different way, resulting in an explicit construction of complementary matchings for arbitrary ℓ and limited values of q. The case of $q = 2$ is simply the binary antipodal matching of Subsection II-A, since it inherently has the complementary property. For $q > 2$, the idea of the construction is as follows.

Assume for now that q is even.[1] Define the mapping $\mathsf{p} : \mathcal{Q} \to \mathcal{B}$ by $\mathsf{p}(x) = 0$ if $x \in [q/2]$ and $\mathsf{p}(x) = 1$ otherwise and, for $z \in [q/2]$, let p_z be the restriction of p

[1] The case of odd q is treated similarly, except in this case the "center" symbol $(q-1)/2$ in $\mathcal{Q}$ is always left unchanged by the matching/bijection.

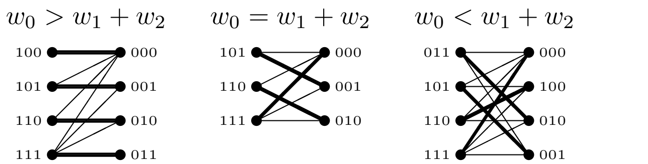

Fig. 3. Graphs $G(\boldsymbol{w})$ for $q = 6$, with highlighted perfect matchings h.

to the domain $\{z, \overline{z}\}$ (thus p_z^{-1} is well defined). Extend p_z to map words over $\{z, \overline{z}\}$ to binary words by applying the mapping componentwise. Also, for $\boldsymbol{x} = (x_i)_{i \in [\ell]} \in \mathcal{Q}^\ell$ and $z \in [q/2]$, let $\mathcal{I}_z = \mathcal{I}_z(\boldsymbol{x}) = \{i : x_i \in \{z, \overline{z}\}\}$. The notation $\boldsymbol{x}_{|z}$ will indicate the subword of $\boldsymbol{x}$ that is indexed by $\mathcal{I}_z$.

For $q > 2$, the value of the bijection $\varphi_{\mathsf{C}}(\boldsymbol{x})$ underlying the complementary matching will be obtained by selectively applying the bijection $\mathsf{f}_z = \mathsf{p}_z^{-1} \circ \varphi_{\mathsf{B}} \circ \mathsf{p}_z$ to the various subwords $\boldsymbol{x}_{|z}$. The decision to apply f_z to any subword $\boldsymbol{x}_{|z}$ will depend on the sign of $\mathsf{c}(\boldsymbol{x}_{|z})$ and on the result of yet another bijection $\mathsf{h} = \mathsf{h}_{\boldsymbol{w}}$, namely the one underlying a perfect matching guaranteed in Proposition 2.4 for the bipartite graph $G(\boldsymbol{w})$ therein, with $\boldsymbol{w}$ equal to the vector $\boldsymbol{w}(\boldsymbol{x})$ defined by

$$(\boldsymbol{w}(\boldsymbol{x}))_z = |\mathsf{c}(\boldsymbol{x}_{|z})| \, , \quad z \in [q/2] \; .$$

Specifically, for this $\boldsymbol{w}$, fix such a perfect matching of $G(\boldsymbol{w})$ and let $\boldsymbol{a}(\boldsymbol{x}) = (a_z)_{z \in [q/2]}$ be a binary vector with $a_z = 0$ if $\mathsf{c}(\boldsymbol{x}_{|z}) \le 0$ and $a_z = 1$ otherwise. It is not hard to see that $\boldsymbol{x} \in \mathcal{L}_{\ell,q}$ if and only if $\boldsymbol{a}(\boldsymbol{x}) \in L(\boldsymbol{w})$ in the graph $G(\boldsymbol{w})$. Writing $\boldsymbol{b}(\boldsymbol{x}) = \mathsf{h}_{\boldsymbol{w}(\boldsymbol{x})}(\boldsymbol{a}(\boldsymbol{x}))$, we now define the mapping $\varphi_{\mathsf{C}} : \mathcal{L}_{\ell,q} \to \mathcal{R}_{\ell,q}$ as follows: for every $\boldsymbol{x} \in \mathcal{L}_{\ell,q}$, the image $\boldsymbol{y} = \varphi_{\mathsf{C}}(\boldsymbol{x})$ satisfies $\mathcal{I}_z(\boldsymbol{y}) = \mathcal{I}_z(\boldsymbol{x})$ for every $z \in [q/2]$, and the subword $\boldsymbol{y}_{|z}$ is given by

$$\boldsymbol{y}_{|z} = \begin{cases} \boldsymbol{x}_{|z} & \text{if } (\boldsymbol{a}(\boldsymbol{x}))_z = 0 \text{ or } (\boldsymbol{b}(\boldsymbol{x}))_z = 1 \\ \mathsf{f}_z(\boldsymbol{x}_{|z}) & \text{otherwise} \end{cases} \; .$$

Proposition 2.7: For the mapping φ_{C} so defined, the set $\{(\boldsymbol{x}, \varphi_{\mathsf{C}}(\boldsymbol{x})) : \boldsymbol{x} \in \mathcal{L}_{\ell,q}\}$ is a complementary matching in $\mathcal{G}_{\ell,q}$.

The proof is straightforward and relies on the fact that $\boldsymbol{w}(\boldsymbol{x})$ and $\boldsymbol{b}(\boldsymbol{x})$ can be recovered from $\varphi_{\mathsf{C}}(\boldsymbol{x})$ and, hence, so can $\boldsymbol{a}(\boldsymbol{x})$ by the bijectivity of $\mathsf{h}_{\boldsymbol{w}(\boldsymbol{x})}$. To complete the above construction of φ_{C}, we must explicitly specify the bijection $\mathsf{h}_{\boldsymbol{w}}$ for each possible $\boldsymbol{w}$. In principle, the variety of mappings is bounded for any fixed q (i.e., is independent of ℓ for ℓ sufficiently large), since there are finitely many distinct graphs $G(\boldsymbol{w})$ that can be induced by different $\boldsymbol{w}$. For one thing, given a $\boldsymbol{w}$, the graphs induced by all $\tilde{\boldsymbol{w}}$ which are permutations of $\boldsymbol{w}$ are isomorphic (through a permutation of indices of the words representing the vertices). It thus suffices to consider $\boldsymbol{w}$ satisfying $w_0 \ge w_1 \ge \ldots \ge w_{q/2-1}$. Even so, the number of the induced graphs grows rather quickly with q. The vertex sets of such graphs are closely related to the self-dual equivalence classes of Boolean threshold functions of $q/2$ or fewer variables (see, e.g., [1]; here, we distinguish cases when the threshold can be met with equality). For limited q (the sequence in [1] is given up to $q/2 = 10$), these graphs, along with the corresponding perfect matching/bijection h, can be computed explicitly. For example, in the case of $q = 6$, there are three different graphs that can arise depending on relations between the entries of $\boldsymbol{w} = (w_0 \, w_1 \, w_2)$. The relations, the graphs, and the corresponding matchings for this case are shown in Figure 3.

Remark 2.1: The above construction essentially solves the matching problem for $\mathcal{G}_{\ell,q}$, with small q and arbitrary ℓ, by reducing it to one for $\mathcal{G}_{\ell',q'}$ with small ℓ' (namely, $q/2$) and large q' (proportional to ℓ). For example, the rightmost graph of Figure 3, which arises when solving for $\mathcal{G}_{\ell,6}$, is the same as the graph in Figure 2, which arose when solving for $\mathcal{G}_{3,4}$. Thus, there appears to be a duality between these regimes. ■

Input: Sequence u of $kn^2{-}2n{+}1$ information bits.

1) With each k successive bits interpreted as an element of $\mathcal{Q}$, arrange first $k(n{-}1)^2$ bits of u into upper-left $(n{-}1){\times}(n{-}1)$ subarray of $A = (A_{i,j})_{i,j \in [n]}$.
2) With each $k{-}1$ successive bits interpreted as an element of $[q/2]$, arrange remaining $(k{-}1)(2n{-}1)$ bits of u into row $n{-}1$ and column $n{-}1$ of A ($2n{-}1$ elements).
3) If A has any rows with positive cost then flip these rows.
4) If A has any columns with positive cost then flip these columns.
5) If flips happened in **Step 4** then go to **Step 3**, otherwise terminate and output the final array.

Output: $n \times n$ array A over the alphabet $\mathcal{Q}$.

Fig. 4. Encoding algorithm for iterative flipping code, for $q = 2^k$.

III. Row–column cost limiting codes

In this section, we apply the q-ary antipodal matchings of Section II to the problem of encoding data into $\mathcal{A}_{n \times n}$, as described in Section I. We begin with a generalization of the iterative flipping code presented in [10] to the case of $q > 2$. As in [10], complexity considerations of this scheme will motivate alternatives based on q-ary antipodal matchings. In this section, for simplicity, we will assume that $q = 2^k$, $k > 1$.

A. Iterative flipping code

The q-ary iterative flipping code for $q = 2^k$ encodes $kn^2{-}2n{+}1$ information bits into $n{\times}n$ arrays belonging to $\mathcal{A}_{n \times n}$. Encoding proceeds as in Figure 4. The *flip* operation in the algorithm corresponds to applying the word-complementation mapping $\boldsymbol{x} \mapsto \overline{\boldsymbol{x}}$ to the respective rows and/or columns.

The encoding procedure is guaranteed to terminate since each row or column flip strictly reduces the sum of all array entries. Since the procedure terminates only when no row or column has positive cost, the final array must belong to $\mathcal{A}_{n \times n}$.

Let $U = (U_{i,j})_{i,j \in [n]}$ denote the array formed after Steps 1–2 in Figure 4. The array U consists of the input bits u interpreted as elements of $\mathcal{Q} = [q]$ and $[q/2]$, and we shall refer to it as the input array. In a manner similar to the binary case in [10], the input array U can be decoded as follows from the encoded array A.

Proposition 3.1: If U and A are respectively the input array and the encoded array from the algorithm of Figure 4, then

$$U_{i,j} = \begin{cases} U_{i,j} & \text{if } \mathsf{p}(A_{i,n-1}) \oplus \mathsf{p}(A_{n-1,j}) \oplus \mathsf{p}(A_{n-1,n-1}) = 0 \\ \overline{U}_{i,j} & \text{otherwise} \end{cases}$$

Input: For n even, arbitrary sequence u of kn^2-2n-4 bits.

1) Using first $k((n-1)^2-1)$ bits of u, for $i,j \in [n-1]$ except $i=j=n-2$, set $A_{i,j}$ to symbols of $\mathcal{Q}$.
2) Using next $2(k-1)(n-2)$ bits of u, fill $A_{[n-2],n-1}$ and $A_{n-1,[n-2]}$ with *even* symbols of $\mathcal{Q}$.
3) Using remaining $4(k-2)$ bits of u, for $i,j \in \{n-2,n-1\}$, set $A_{i,j}$ to symbols of $[q/4]$.
4) Flip all of the first $n-2$ rows of A having positive cost.
5) For all $j \in [n-2]$ do:
   If $\mathsf{c}(A_{[n],j}) \ \% \ 4 = 0$, set $A_{n-1,j} \mathrel{+}= 1$.
   If (the resulting) $\mathsf{c}(A_{[n],j})$ is positive then do:
   a) replace $A_{[n],j}$ with $\varphi_{\mathsf{U}}(A_{[n],j})$;
   b) set $A_{n-1,j} = (A_{n-1,j}+1) \ \% \ q$.
6) For $j \in \{n-2,n-1\}$ do:
   a) if $\mathsf{c}(A_{[n-2],j}) > 0$, replace $A_{[n-2],j}$ with $\varphi_{\mathsf{U}}(A_{[n-2],j})$ and set $A_{n-2,j} \mathrel{+}= q/4$;
   b) if $\mathsf{c}(A_{j,[n-2]}) > 0$, replace $A_{j,[n-2]}$ with $\varphi_{\mathsf{U}}(A_{j,[n-2]})$ and set $A_{n-1,j} \mathrel{+}= q/4$.

Output: The resulting $n \times n$ array A.

Fig. 5. Encoding algorithm for q-ary antipodal matching code that is based on unary matching, for $q = 2^k$.

where $\mathsf{p}(\cdot)$ is as defined in Subsection II-D and "$\oplus$" denotes addition modulo 2 ("exclusive-OR").

Thus, as in the binary case, the number of operations required for decoding is linear in the number of array entries. The true complexity of encoding, on the other hand, is less clear. The best guarantee we have presently is an upper bound of $O(qn^2)$ row–column flips (equivalently, $O(qn^3)$ symbol flips) and iterations between rows and columns, which follows readily from the above-noted fact that each row–column flip strictly reduces the sum of array entries by at least 1. The complexity guarantee thus grows linearly with q, which could be problematic in practical applications.

B. Antipodal matching code

In this subsection, we generalize the two-dimensional antipodal matching code from [10] to the case of $q > 2$, making use of the q-ary antipodal matching constructions of Section II. The key idea is that the underlying q-ary antipodal matching (unary or complementary) permits iteration-free encoding by not creating new constraint violations, unlike the row–column flips of the iterative flipping encoder.

The unary and complementary matchings of Section II give rise to slightly different antipodal matching code constructions, with the difference being primarily in how redundancy is incorporated to allow for decodability. We shall provide the details only for the unary matching. Also, we shall confine our description to even n and (again) $q = 2^k$, though the basic idea applies generally.

For these choices, a key property of the unary q-ary antipodal matching is that the bijection $\varphi_{\mathsf{U}}(\boldsymbol{x})$ preserves the parity of the sum $\sum_i x_i$; equivalently, it preserves the remainder of the cost $\mathsf{c}(\boldsymbol{x})$ modulo 4. This will be the means for incorporating the aforementioned redundancy. Figures 5 and 6 depict the encoding and decoding algorithms for the antipodal matching code that is based on unary matching. In these figures, the operation "%" stands for remaindering, and the *flip* operation is the same as in Subsection III-A.

Input: $n \times n$ array A.

1) For $j \in \{n-2,n-1\}$ do:
   a) if $A_{n-2,j} \geq q/4$, replace $A_{[n-2],j}$ with $\varphi_{\mathsf{U}}^{-1}(A_{[n-2],j})$ and set $A_{n-2,j} \mathrel{-}= q/4$;
   b) if $A_{n-1,j} \geq q/4$, replace $A_{j,[n-2]}$ with $\varphi_{\mathsf{U}}^{-1}(A_{j,[n-2]})$ and set $A_{n-1,j} \mathrel{-}= q/4$.
2) For all $j \in [n-2]$ do:
   If $\mathsf{c}(A_{[n],j}) \ \% \ 4 = 0$ then do:
   a) set $A_{n-1,j} = (A_{n-1,j}-1) \ \% \ q$;
   b) replace $A_{[n],j}$ with $\varphi_{\mathsf{U}}^{-1}(A_{[n],j})$.
   If $A_{n-1,j}$ is odd, set $A_{n-1,j} \mathrel{-}= 1$.
3) Flip all of the first $n-2$ rows of A for which $A_{i,n-1}$ is odd.

Output: Sequence u' of kn^2-2n-4 bits obtained by applying inverse of Encoder Steps 1–3 to corresponding locations in A.

Fig. 6. Decoding algorithm for q-ary antipodal matching code that is based on unary matching, for $q = 2^k$.

Proposition 3.2: The output array A of the encoder of Figure 5 belongs to $\mathcal{A}_{n \times n}$.

Proposition 3.3: Provided that the input to Figure 6 is an array A that was generated as output in Figure 5, the kn^2-2n-4 bits in the output u' in Figure 6 coincide with the corresponding input bits u in Figure 5.

Referring to the encoding algorithm of Figure 5, we see that the unary q-ary antipodal matching bijection φ_{U} may be applied to cost constraint violating (partial) columns and rows (e.g., Step 5a). By the domination property of the matching, namely, $\varphi_{\mathsf{U}}(\boldsymbol{x}) \leq \boldsymbol{x}$, no new (row) constraint violations are induced by these steps. Whether φ_{U} has been applied to a particular column is encoded, in some cases (e.g., Step 5b), in the parity of the sums of the corresponding columns, exploiting the aforementioned parity preserving property of φ_{U}.

References

[1] *The On-Line Encyclopedia of Integer Sequences,* published electronically at http://oeis.org, 2010, Sequence A001532.

[2] M. Aigner, *Lexicographic matching in Boolean algebras, J. Comb. Theory B,* 14 (1973), 187–194.

[3] N.L. Biggs, *Discrete Mathematics,* Oxford Univ. Press, Oxford, 1985.

[4] N. de Bruijn, C. Tengbergen, D. Kruyswijk, *On the set of divisors of a number, Nieuw Arch. Wiskunde,* 23 (1951), 191–193.

[5] N. Dershowitz, S. Zaks, *The Cycle Lemma and some applications, Europ. J. Comb.,* 11 (1990), 35–40.

[6] A. Dvoretzky, Th. Motzkin, *A problem of arrangements, Duke Math. J.,* 14 (1947), 305–313.

[7] C. Greene, D.J. Kleitman, *Strong versions of Sperner's Theorem, J. Comb. Th. A,* 20 (1976), 80–88.

[8] J. Marica, J. Schönheim, *Differences of sets and a problem of Graham, Canad. Math. Bull.,* 12 (1969), 635–637.

[9] E. Ordentlich, G.M. Ribeiro, R.M. Roth, G. Seroussi, P.O. Vontobel, *Coding for limiting current in memristor crossbar memories, 2nd Annual Non-Volatile Memories Workshop,* UCSD, La Jolla, CA, March 2011. Presentation slides accessible at: `http://nvmw.ucsd.edu/2011/`

[10] E. Ordentlich, R.M. Roth, *Low complexity two-dimensional weight-constrained codes, Proc. 2011 IEEE Intl. Symp. Inform. Theory,* St. Petersburg, Russia (Aug. 2011), 116–120.

[11] D.B. Strukov, R.S. Williams, *Four-dimensional address topology for circuits with stacked multilayer crossbar arrays, Proc. Nat'l. Acad. Sci.,* 106 (2009), 20155–20158.