STRUCTURE OF NON-BINARY REGULAR LDPC CYCLE CODES

Jie Huang, Shengli Zhou, and Peter Willett

Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT, 06269

ABSTRACT

In this paper, we study non-binary regular LDPC cycle codes whose parity check matrix has fixed column weight 2 and fixed row weight d. We prove that the parity check matrix of any regular cycle code can be put into a concatenation form of row-permuted block-diagonal matrices after row and column permutations if d is even, or, if d is odd and the code's associated graph contains at least one spanning subgraph that consists of disjoint edges. Utilizing this structure enables parallel processing in linear-time encoding, and parallel processing in sequential belief-propagation decoding, which increases the throughput without compromising performance or complexity. Numerical results are presented to compare the code performance and the decoding complexity.

Index Terms— Nonbinary, LDPC, cycle code, Galois field, graph theory

1. INTRODUCTION

Gallager's binary low-density parity-check (LDPC) codes [1] are excellent error-correcting codes that achieve performance close to the benchmark predicted by the Shannon capacity [2]. The extension of LDPC to non-binary Galois field GF(q) was first investigated empirically by Davey and Mackay over the binary-input AWGN channel [3]. Since then, nonbinary LDPC codes have been actively studied.

The LDPC codes with column weight $j = 2$ in their parity check matrix $\mathbf{H}$ are termed as cycle codes [4]. Although the distance properties of binary cycle codes are not as good as the LDPC codes of column weight $j \geq 3$ [1], it has been shown in [5] that cycle GF(q) codes can achieve near-Shannon-limit performance as q increases. Further, numerical results in [5] demonstrate that cycle GF(q) codes can outperform other LDPC codes, including degree-distribution-optimized binary irregular LDPC codes. For high order fields $q \geq 64$, the best GF(q)-LDPC codes decoded by belief propagation (BP) should be *ultra sparse* [3], with a good example being the cycle codes that have $j = 2$. Reduced complexity algorithms for decoding a general LDPC code over GF(q) have been proposed in [6], [7]. A universal linear-complexity encoding algorithm for any cycle GF(q) code is available in [8]. With the performance and implementation advantages, cycle GF(q) codes are very promising for practical applications.

In this paper, we study LDPC cycle codes whose check matrix has fixed row weight d, termed as d-regular cycle codes. Using graph theory, we prove that through row and column permutations the parity check matrix $\mathbf{H}$ of d-regular cycle GF(q) codes can always be put into *a concatenation form of row-permuted block-diagonal matrices* if d is even, or, if d is odd and the code's associated graph contains at least one spanning subgraph that consists of disjoint edges. This equivalent representation brings several benefits. First, encoding for regular cycle GF(q) codes can be performed *in parallel* in linear time. Second, it enables *parallel processing* in sequential belief propagation decoding for regular cycle GF(q) codes, which improves the decoding throughput considerably without compromising the performance and complexity. It also reduces the storage of the check matrix $\mathbf{H}$ for encoding and decoding, and facilitates code design [13]. Simulation results confirm very good performance and reduced decoding complexity of regular cycle GF(q) codes.

2. MAIN RESULTS ON CODE STRUCTURE

A cycle GF(q) code is an LDPC code whose $m \times n$ parity check matrix $\mathbf{H}$ has weight $j = 2$ for each column. As such, it can be represented by an associated graph $G = (V, E)$ with m vertices $V = \{v_1, \ldots, v_m\}$ and n edges $E = \{e_1, \ldots, e_n\}$, where each vertex represents a constraint node corresponding to a row of $\mathbf{H}$, and each edge represents a variable node corresponding to a column of $\mathbf{H}$ [8].

If the cycle GF(q) code also has a fixed row weight d in $\mathbf{H}$, the graph G is d-regular in that each vertex is exactly linked to d edges [9]. We call this code as regular cycle GF(q) code. Obviously we have $2n = dm$ for regular cycle GF(q) codes.

We first introduce two definitions from graph theory [9].

- k-**factor**: A k-regular spanning subgraph of G that contains all the vertices is called a k-factor of G.

  Obviously, a 1-factor is a spanning subgraph that consists of disjoint edges, while a 2-factor is a spanning subgraph that consists of disjoint cycles.
- k-**factorable**: a graph G is k-factorable if there are edge-disjoint k-factors $G_1, G_2, \ldots, G_L$ such that $G = G_1 \cup G_2 \cdots \cup G_L$.

This work is supported by the ONR grant N00014-07-1-0429 and the NSF grant ECCS-0725562.

For a subgraph G' of G, let $\mathbf{H}_{G'}$ be the sub-matrix of $\mathbf{H}$ restricted to the rows and columns indexed by the vertices and edges of G' respectively, which can be obtained from $\mathbf{H}$ by deleting the rows and columns other than those corresponding to the vertices and edges of $G^{'}$ respectively. Let us now introduce two sub-matrices of $\mathbf{H}$ associated with an edge and a cycle of the graph G. For each edge, the sub-matrix is

$$\tilde{\mathbf{h}}^e = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \tag{1}$$

where α and β correspond to those two nonzero entries of the column of $\mathbf{H}$ indexed by this edge. For a length-k cycle C that consists of k consecutive edges $e_1, e_2, \ldots, e_k$, we can define a $k \times k$ matrix as

$$\tilde{\mathbf{H}}^c = \begin{bmatrix} \alpha_1 & 0 & 0 & \ldots & \beta_k \\ \beta_1 & \alpha_2 & 0 & \ldots & 0 \\ 0 & \beta_2 & \alpha_3 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & \beta_{k-1} & \alpha_k \end{bmatrix}, \tag{2}$$

where α_is and β_is correspond to those two nonzero entries of the column of $\mathbf{H}$ indexed by edge e_i.

If a matrix $\mathbf{H}_1$ can be transformed into another matrix $\mathbf{H}_2$ simply through row and column permutations, we deem $\mathbf{H}_1$ equivalent to $\mathbf{H}_2$ and denote this relationship as $\mathbf{H}_1 \cong \mathbf{H}_2$.

Our main results are the following.

Theorem 1 *For a cycle GF(q) code, if its associated graph G is d-regular with $d = 2r$, its parity check matrix $\mathbf{H}$ of size $m \times n$ has the equivalent form*

$$\mathbf{H} \cong [\bar{\mathbf{H}}_1, \mathbf{P}_2\bar{\mathbf{H}}_2, \ldots, \mathbf{P}_r\bar{\mathbf{H}}_r], \tag{3}$$

where $\mathbf{P}_i$ is $m \times m$ permutation matrix, and $\bar{\mathbf{H}}_i$ is of size $m \times m$, $1 \le i \le r$. The matrix $\bar{\mathbf{H}}_i$ has an equivalent block-diagonal form

$$\bar{\mathbf{H}}_i \cong \mathrm{diag}(\tilde{\mathbf{H}}^c_{i,1}, \tilde{\mathbf{H}}^c_{i,2}, \ldots, \tilde{\mathbf{H}}^c_{i,L_i}), \tag{4}$$

where the matrix $\tilde{\mathbf{H}}^c_{i,l}$ has the form of (2) *and is of size $k_{i,l} \times k_{i,l}$ that satisfies $m = \sum_{l=1}^{L_i} k_{i,l}$.*

Theorem 2 *Consider a regular cycle GF(q) code with $d = 2r + 1$. If its associated graph G contains at least one 1-factor, then its parity check matrix $\mathbf{H}$ of size $m \times n$ has the equivalent form*

$$\mathbf{H} \cong [\bar{\mathbf{H}}_1, \mathbf{P}_2\bar{\mathbf{H}}_2, \ldots, \mathbf{P}_r\bar{\mathbf{H}}_r, \mathbf{P}^e\bar{\mathbf{H}}^e] \tag{5}$$

where $\mathbf{P}_i$s and $\mathbf{P}^e$ are permutation matrices, $\bar{\mathbf{H}}_i$ is an $m \times m$ block-diagonal matrix having the form as in (4), $i = 1, \ldots, r$, $\bar{\mathbf{H}}^e$ is an $m \times \frac{m}{2}$ matrix having an equivalent block-diagonal form as

$$\bar{\mathbf{H}}^e \cong \mathrm{diag}(\tilde{\mathbf{h}}^e_1, \tilde{\mathbf{h}}^e_2, \ldots, \tilde{\mathbf{h}}^e_{\frac{m}{2}}), \tag{6}$$

where $\tilde{\mathbf{h}}^e_i$ is a vector having the form as in (1)*.*

Proof of Theorem 1: If G is d-regular with $d = 2r$, $r > 0$, G is 2-factorable as can be inferred from Corollary 2.1.5 of [9, p.33]. Denote the r edge-disjoint 2-factors of G by $G_1, G_2, \ldots, G_r$. Arrange the columns of $\mathbf{H}$ in such a pattern that the columns indexed by the edges of G_1 are placed in the first m columns, followed by the m columns indexed by the edges of G_2 until the m columns which are indexed by the edges of G_r. This way, $\mathbf{H}$ is partitioned to r sub-matrices of size $m \times m$ each. Arranged as $\mathbf{H} \cong [\mathbf{H}_{G_1}, \ldots, \mathbf{H}_{G_r}]$, where $\mathbf{H}_{G_i}$ is the sub-matrix of $\mathbf{H}$ associated with G_i.

Now we show that each $m \times m$ sub-matrix $\mathbf{H}_{G_i}$ has an equivalent block diagonal form as in (4). Each 2-factor G_i can be decomposed into a set of disjoint cycles. Suppose G_i consists of L_i disjoint cycles $C_{i,l}$, $1 \le l \le L_i$, where $C_{i,l}$ is of length $k_{i,l}$ that satisfy $m = \sum_{l=1}^{L_i} k_{i,l}$. Arrange the rows and columns of $\mathbf{H}_{G_i}$ in sequence of rows and columns indexed by $C_{i,1}, C_{i,2}, \ldots, C_{i,L_i}$, the resultant matrix will have a block-diagonal form $\mathrm{diag}(\tilde{\mathbf{H}}^c_{i,1}, \tilde{\mathbf{H}}^c_{i,2}, \ldots, \tilde{\mathbf{H}}^c_{i,L_i})$, where $\tilde{\mathbf{H}}^c_{i,l}$ represents the matrix associated with $C_{i,l}$ and has a form as in (2). Thus we have $\mathbf{H}_{G_i} = \mathbf{P}_i\bar{\mathbf{H}}_i\mathbf{R}_i$, where $\bar{\mathbf{H}}_i$ is defined in (4), and $\mathbf{P}_i$ and $\mathbf{R}_i$ are permutation matrices, $1 \le i \le r$.

Therefore, the matrix $\mathbf{H}$ can be arranged to have an equivalent form $[\mathbf{P}_1\bar{\mathbf{H}}_1\mathbf{R}_1, \mathbf{P}_2\bar{\mathbf{H}}_2\mathbf{R}_2, \ldots, \mathbf{P}_r\bar{\mathbf{H}}_r\mathbf{R}_r]$, We can further permute the rows of $\mathbf{H}$ to let $\mathbf{P}_1$ be the identity matrix and permute the columns of $\mathbf{H}_{G_i}$ to let each $\mathbf{R}_i$ be the identity matrix. The resultant matrix would have a form like (3). This completes the proof. ■

Proof of Theorem 2 is omitted here due to space limitations (available in [13]).

3. ENCODING AND DECODING BENEFITS

The code structure in Section 2 brings many benefits on encoding, decoding, storage, and design of regular cycle codes, as discussed in [13]. Due to space limitations, we here detail the benefits to the encoding and decoding.

3.1. Linear-time encoding in parallel

The representation in Theorems 1 and 2 enables efficient encoding as follows. For $d = 2r$, partition the codeword $\mathbf{x}$ into r sub-codewords of size m as $\mathbf{x} = [\mathbf{x}^T_{c,1}, \mathbf{x}^T_{c,2}, \ldots, \mathbf{x}^T_{c,r}]^T$. For $d = 2r + 1$, partition the codeword $\mathbf{x}$ into $r + 1$ sub-codewords as $\mathbf{x} = [\mathbf{x}^T_{c,1}, \mathbf{x}^T_{c,2}, \ldots, \mathbf{x}^T_{c,r}, \mathbf{x}^T_e]^T$, where $\mathbf{x}_{c,i}$ is of size m, $1 \le i \le r$, and $\mathbf{x}_e$ is of size $m/2$. Without loss of generality, assume $\bar{\mathbf{H}}_1$ is full rank; let $\mathbf{x}_{c,1}$ contain the parity symbols and the rest of $\mathbf{x}$ contain information symbols, which leads to a code rate of $(d-2)/d$. A valid codeword satisfies $\mathbf{Hx} = \mathbf{0}$, which implies that

$$\bar{\mathbf{H}}_1\mathbf{x}_{c,1} = \begin{cases} -\mathbf{P}^c_2\bar{\mathbf{H}}_2\mathbf{x}_{c,2} \cdots - \mathbf{P}^c_r\bar{\mathbf{H}}_r\mathbf{x}_{c,r}, & d = 2r \\ -\mathbf{P}^c_2\bar{\mathbf{H}}_2\mathbf{x}_{c,2} \cdots - \mathbf{P}^c_r\bar{\mathbf{H}}_r\mathbf{x}_{c,r} - \mathbf{P}^e\bar{\mathbf{H}}^e\mathbf{x}_e, & d = 2r+1 \end{cases} \tag{7}$$

From (4), $\bar{\mathbf{H}}_1$ is block diagonal diag$(\tilde{\mathbf{H}}^c_{1,1}, \ldots, \tilde{\mathbf{H}}^c_{1,L_1})$. According to the sizes of $\{\tilde{\mathbf{H}}^c_{1,l}\}_{l=1}^{L_1}$, let us partition $\mathbf{x}_{c,1}$ and the right-hand side of (7) into L_1 pieces as $[\mathbf{x}^T_{c,1,1}, \ldots, \mathbf{x}^T_{c,1,L_1}]^T$ and $[\mathbf{b}^T_1, \ldots, \mathbf{b}^T_{L_1}]^T$, respectively. Computation of $\mathbf{x}_{c,1}$ requires solving the following L_1 equations

$$\tilde{\mathbf{H}}^c_{1,i}\mathbf{x}_{c,1,i} = \mathbf{b}_i, \quad 1 \le i \le L_1. \tag{8}$$

A linear time algorithm for solving these equations has been proposed in Lemma 4 of [8]. Note that solving these L_1 equations can be performed in parallel, thus encoding can be performed *in parallel* in linear time. This provides a lot of flexibility in the implementation of efficient encoders, which is quite desirable especially when the codeword length is large. Note that the universal linear-time encoding algorithm presented in [8] can only work in a serial manner.

3.2. Parallel processing in sequential BP decoding

Recently, a fully sequential version of standard belief propagation (BP) decoding has been proposed to speed up the convergence of decoding, which is denoted as shuffled BP in [10] and sequential updating schedule in [11]. Compared with standard BP decoding which works in a fully parallel manner, sequential BP decoding works in a column-by-column manner. It has been shown through simulations that the average number of iterations of the sequential BP algorithm can be about half that of the parallel BP algorithm, where parallel BP and sequential BP decoding achieve similar error performance [10, 11]. The complexity per iteration for both algorithms is similar, resulting in a lower total complexity for the sequential BP algorithm [10, 11].

To decrease the decoding delay of the sequential BP and preserve the parallelism advantages of the parallel BP, a partially parallel decoding scheme named "group shuffled BP" is developed in [10]. In the group shuffled BP algorithm, the columns of $\mathbf{H}$ are divided into a number of groups. In each group, the updating of messages is processed in parallel, but the precessing of groups remains sequential. If the number of groups is one, group shuffled BP reduces to the parallel BP algorithm. If the number of groups equals the number of columns of $\mathbf{H}$, group shuffled BP reduces to the sequential BP algorithm. Thus, group shuffled BP (partially parallel BP) algorithm offers better throughput/complexity tradeoffs in the implementation of efficient decoders.

With respect to the sequential BP algorithm, if there are consecutive columns of $\mathbf{H}$ which are orthogonal to each other, i.e., no two columns intersect at a common row, then the updating for these columns can be carried out simultaneously. By performing updating for consecutive orthogonal columns simultaneously, we can improve the throughput of sequential BP algorithm without any penalty in error performance or total decoding complexity. We denote this algorithm as sequential BP decoding with parallel processing. This is analogous in principle to a partially parallel BP algorithm where the columns in each group are orthogonal.

For a cycle GF(q) code, a collection of columns of $\mathbf{H}$ are orthogonal if and only if their corresponding edges in its associated graph G are independent. With the structures presented in Section 2, it is easy to find orthogonal columns for regular cycle GF(q) codes. We find that for any regular cycle GF(q) code the columns of $\mathbf{H}$ can be partitioned into at most $\frac{3}{2}d$ orthogonal groups (See more details in [13]).

Compared with sequential BP decoding, which works in a column-by-column manner and takes n steps, by running updating for columns in each orthogonal group simultaneously, we can greatly improve the throughput of sequential BP decoding algorithm for regular cycle GF(q) codes by a factor at least $\frac{2n}{3d}$. Note that n is usually large while d is usually small. The large throughput improvement is very appealing in the implementation of efficient decoders. Note that the performance and complexity advantages of sequential BP decoding are not compromised.

4. SIMULATION RESULTS

In all simulations the codewords are transmitted over AWGN channel with binary phase-shift-keying (BPSK) modulation. For each SNR, we run simulations until more than 40 block errors have been observed or up to $1,000,000$ block decodings. Here we only present the results for the block length of 1008 bits; more simulation results with different block lengths are available in [13].

Test Case 1 (Regular versus irregular cycle GF(q) codes). Fig. 1 compares the performance of regular and irregular cycle GF(q) codes under standard BP decoding up to 80 iterations where the code rate is $1/2$ and the codeword length is 1008 bits. The cycle codes over GF(2^6) have a symbol length of 84 and the cycle codes over GF(2^8) have a symbol length of 63. Also plotted is the performance of a binary irregular rate-$1/2$ LDPC code constructed by the Progressive Edge-Growth algorithm [12] and that of a rate-$1/2$ Mackay's regular-(3,6) code, both having a code length of 1008 bits and decoded by standard BP up to 80 iterations. The binary irregular code has a density-evolution-optimized degree distribution pair achieving an impressive iterative decoding threshold of 0.3347 dB, from Table I in [14].

It has been shown in [5] that irregular cycle codes over GF(q) can outperform binary degree-distribution-optimized LDPC codes. We observe from Fig. 1 that regular cycle codes can also outperform binary degree-distribution-optimized LDPC codes. In fact, Fig. 1 shows that regular cycle codes and irregular cycle codes have similar performance.

Test Case 2 (Sequential versus parallel BP decoding). Figs. 2 and 3 show the comparisons on the error performance and the average number of iterations between the proposed sequential BP decoding with parallel processing and standard BP decoding for those regular cycle GF(q) codes shown in

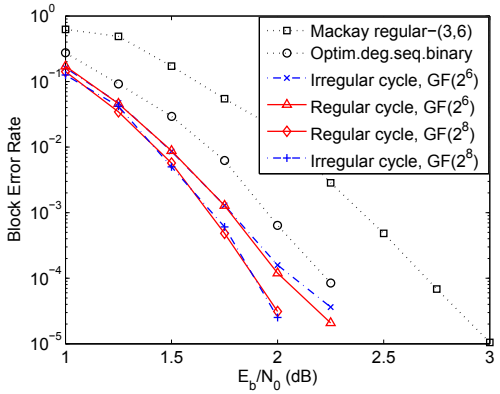

Fig. 1. Performance comparison of irregular and regular cycle codes with binary degree-distribution-optimized irregular LDPC code and the Mackay's (3,6) regular code; The code length is 1008 bits.

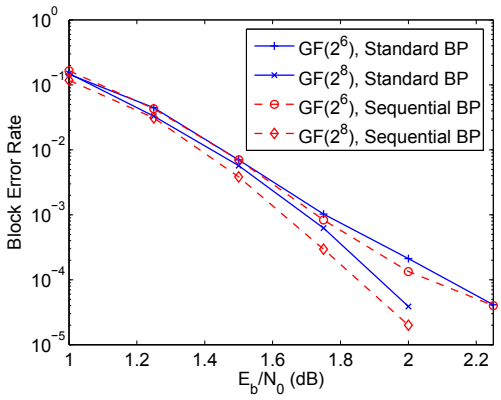

Fig. 2. Performance with sequential and standard BP decodings

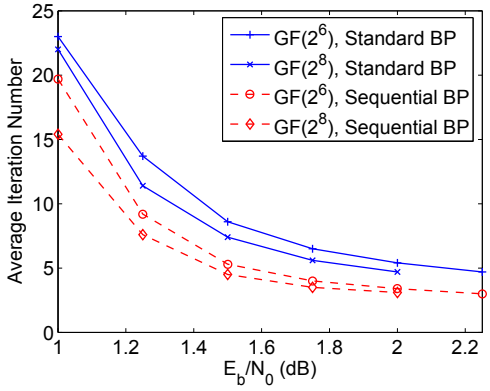

Fig. 3. The average number of iterations of the standard BP decoding and the proposed sequential BP decoding

Fig. 1. The maximum number of iterations is set to be 80. We observe from Fig. 2 that the sequential BP decoding with parallel processing achieves slightly better performance than the standard parallel BP decoding. More importantly, Fig. 3 shows that the average number of iterations for the sequential BP decoding is about 30 percent less than that of the standard BP decoding at high SNR. Hence, the total decoding complexity for the proposed algorithm is 30 percent less than that for standard BP decoding algorithm. Moreover, the proposed parallel processing enables a speedup on the throughput of sequential BP decoding by a factor at least $\frac{2n}{3d} = 10.5$ for the regular GF(2^8) code and at least $\frac{2n}{3d} = 14$ for the regular GF(2^6) codes.

5. CONCLUSIONS

Through graph-theoretic analysis, we presented an equivalent concatenation form of row-permuted block-diagonal matrices for the parity check matrix of non-binary regular LDPC cycle codes. Encoding utilizing this form can be performed in parallel in linear time. Decoding utilizing this form enables parallel processing in sequential BP decoding, which considerably increases the decoding throughput without compromising performance or complexity. Simulations confirm that regular cycle GF(q) codes have very good performance.

6. REFERENCES

[1] R. G. Gallager, *Low Density Parity Check Codes,* Cambridge, MA: MIT Press, 1963.

[2] D. J. C. Mackay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, Mar. 1999.

[3] M. C. Davey and D. Mackay, "Low-density parity-check codes over GF(q)," *IEEE Commun. Lett.*, pp. 165 – 167, June 1999.

[4] D. Jungnickel and S. A. Vanstone, "Graphical codes revisited," *IEEE Trans. Inform. Theory*, vol. 43, pp. 136–146, Jan. 1997.

[5] X.-Y. Hu and E. Eleftheriou, "Binary representation of cycle Tanner-graph GF(2^b) codes," *Proc. of ICC*, June 2004.

[6] H. Song and J. R. Cruz, "Reduced-complexity decoding of Q-ary LDPC codes for magnetic recording," *IEEE Trans. Magn.*, vol. 39, pp. 1081–1087, Mar. 2003.

[7] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC codes over GF(2^q)," in *Proc. IEEE Inform. Theory Workshop*, pp. 70 – 73, 2003.

[8] J. Huang and J.-K. Zhu, "Linear time encoding of cycle GF(2^p) codes through graph analysis," *IEEE Commun. Lett.*, vol. 10, pp. 369 – 371, May 2006.

[9] D. Reinhard, *Graph Theory*, 2nd ed., Springer-Verlag, 2000.

[10] J.-T. Zhang and M. P. Fossorier, "Shuffled iterative decoding," *IEEE Trans. Commun.*, vol. 53, pp. 209-213, Feb. 2005.

[11] H. Kfir and I. Kanter, "Parallel versus sequential updating for belief propagation decoding," *Physica A: Statistical Mechanics and its Applications*, vol. 330, pp. 259-270, Dec. 2003.

[12] X.-Y. Hu, E. Eleftheriou and D.-M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. on Inform. Theory*, vol. 51, Jan. 2005.

[13] J. Huang, S. Zhou and P. Willett, "On regular LDPC cycle codes over GF(q)," *IEEE Trans. Inform. Theory*, submitted.

[14] T. Richardson, A. Shokrollahi and R. Urbanke, "Design of provably good low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619-637, Feb. 2001.