Informed Fixed Scheduling for Faster Convergence of Shuffled Belief-Propagation Decoding

Chaudhry Adnan Aslam, *Student Member, IEEE*, Yong Liang Guan, *Member, IEEE*, Kui Cai, *Senior Member, IEEE*, and Guojun Han, *Senior Member, IEEE*

***Abstract*— A novel *informed fixed scheduling* (IFS) scheme for shuffled belief-propagation (BP) decoding of binary low-density parity-check (LDPC) code is introduced to improve the BP decoding convergence. The IFS finds an appropriate order of variable nodes in accordance with the number of updated neighbors in the code graph, ensuring that the maximum number of latest message updates is utilized within a single iteration. This allows the utilization of most reliable message updates in a timely manner, leading to faster error-rate convergence. Simulation results show that the proposed IFS scheme improves the convergence speed of BP decoder by up to 20% for regular LDPC codes and 45% for irregular LDPC codes, without affecting the error-rate performance, at medium-to-high signal-to-noise ratio over binary-input additive white Gaussian noise channel.**

***Index Terms*— LDPC codes, scheduling scheme, belief-propagation decoding, convergence speed, decoding complexity.**

I. Introduction

LDPC codes are known for their capacity approaching performance under the sum-product belief-propagation (BP) decoding algorithm, in which message updates are iteratively exchanged between the variable and check nodes over the bi-partite code graph. A BP schedule defines the order of message-passing over this code graph. The *flooding* [1] schedule is the conventional scheduling method where all the variable-to-check (V2C) and check-to-variable (C2V) message updates are simultaneously exchanged. Instead of flooding, it is also possible to schedule the message-passing in fixed sequential order such that the message updates in the current iteration are produced using the latest available information. The *shuffled* [2] and *layered* [3] scheduling schemes are two commonly used sequential schedules, where variable or check nodes are serially processed according to a predetermined order. Compared to flooding, sequential processing over the code graph improves the convergence speed of BP algorithm. Analytical studies have shown that the sequential schedules converge twice as fast as that of the flooding schedule.

Manuscript received July 30, 2016; revised September 20, 2016; accepted October 10, 2016. Date of publication October 13, 2016; date of current version January 6, 2017. This work was supported by the NTU Research Scholarship, in part by the SUTD-MIT International Design Center (IDC), and in part by the National Natural Science Foundation of China under grant number 61471131. The associate editor coordinating the review of this paper and approving it for publication was C. Jego.

C. A. Aslam and Y. L. Guan are with the School of Electrical and Electronics Engineering, Nanyang Technological University, 639798 Singapore (e-mail: ad0001ry@e.ntu.edu.sg; eylguan@ntu.edu.sg).

K. Cai is with the Department of Science, Singapore University of Technology and Design, 487372 Singapore (e-mail: cai_kui@sutd.edu.sg).

G. Han is with the Guangdong University of Technology, Guangzhou 510006, China (e-mail: gjhan@gdut.edu.cn).

Digital Object Identifier 10.1109/LCOMM.2016.2617309

With regards to sequential schedule, it is very important to select a proper sequence of variable nodes, as it can influence the convergence-rate of BP decoder. In this direction, a *maximum mutual-information* based BP schedule, which predicts the next message update on the basis of mutual-information increase, has been proposed in [4]. Again, this algorithm improves the convergence performance, however the resultant hardware implementation complexity and the storage requirement may be prohibitively large as it has to record the complete sequence of message updates. Recently, a *column-weight* (CW) based fixed scheduling scheme has been reported in [5], in which the variable nodes are sequentially updated following the high-to-low column weight order. By design, the CW based scheduling works very well on irregular LDPC codes, but has inconsequential effect on the convergence of regular LDPC codes. Thus, it is of great interest to devise a scheduling strategy for codes with either equal or unequal column-weight variable nodes. Such scheduling scheme will benefit both the regular and irregular LDPC codes.

In this direction, a novel *informed fixed scheduling (IFS)* scheme is proposed. The objective of the IFS scheme is to improve the BP decoding convergence, thereby saving the power consumption and increasing the decoding throughput. In this scheme, a fixed sequence of variable nodes is determined based on the information of latest message updates. The IFS ensures that the maximum number of such latest message updates are utilized within a single decoding iteration. This scheduled sequence is then serially decoded using the shuffled BP algorithm. It will be shown that the IFS based shuffled BP decoder attains faster error-rate convergence than the conventional shuffled BP decoder in the range of medium-to-high signal-to-noise ratio (SNR), without incurring additional run-time computations or error-rate degradation.

II. Parallel vs. Sequential BP Scheduling

The variable-node based sequential BP schedules introduce the problem of finding a suitable sequence of variable nodes. In order to explain the proposed methodology, let us first observe the difference between the shuffled and flooding BP schedules. Over a bi-partite (N, M) LDPC code graph with N variable nodes, v_n for $1 \leq n \leq N$, and M check nodes, c_m for $1 \leq m \leq M$, the BP algorithm iteratively computes and passes new message updates from variable node v_n to check node c_m, denoted by $V_{v_n \to c_m}$, and from check node c_m to variable node v_n, denoted by $C_{c_m \to v_n}$, along the edges of the code graph. These message updates can be

computed as [1]

$$V_{v_n \to c_m}^{(i)} = L_{v_n} + \sum_{c_j \in \mathcal{M}(v_n) \setminus c_m} C_{c_j \to v_n}^{(i)} \tag{1}$$

$$C_{c_m \to v_n}^{(i)} = 2 \tanh^{-1} \left(\prod_{v_j \in \mathcal{N}(c_m) \setminus v_n} \tanh \left(\frac{V_{v_j \to c_m}^{(i-1)}}{2} \right) \right) \tag{2}$$

where i is the iteration count, L_{v_n} is the intrinsic channel information for variable node v_n, $\mathcal{M}(v_n) \setminus c_m$ is the set of neighbors of variable node v_n except check node c_m and $\mathcal{N}(c_m) \setminus v_n$ is the set of neighbors of check node c_m except variable node v_n.

Under the flooding schedule, the message propagation takes place simultaneously. In contrast, the shuffled schedule updates variable nodes serially. For better understanding, we split the C2V update expression (2) into two factors, given by

$$C_{c_m \to v_n}^{(i)} = 2 \tanh^{-1} \left(\prod_{\substack{v_j \in \mathcal{N}(c_m) \\ [-2pt] v_j < v_n}} \tanh \left(\frac{V_{v_j \to c_m}^{(i)}}{2} \right) \cdot \prod_{\substack{v_j \in \mathcal{N}(c_m) \\ v_j > v_n}} \tanh \left(\frac{V_{v_j \to c_m}^{(i-1)}}{2} \right) \right) \tag{3}$$

In this expression, the first term incorporates the V2C messages computed in the current iteration i, while the second one takes on the V2C messages computed in the previous iteration $i-1$. The shuffled decoder makes it possible to access the latest message updates $V_{v_j \to c_m}^{(i)}$ for $v_j < v_n$ via check nodes c_m for $c_m \in \mathcal{M}(v_n)$ while updating variable node v_n. This is how the shuffled BP decoder yields faster convergence as compared to the flooding schedule. Therefore, it makes sense to select a variable node such that its neighbouring check nodes are connected with maximum number of updated variable nodes. In this way, a large number of variable nodes will receive the latest message updates within a single decoding iteration, consequently reducing the iteration count required for error-rate convergence. Thus, instead of arbitrary variable node selection, scheduling based on the availability of latest message updates is adopted in the proposed IFS scheme.

III. Informed Fixed Scheduling (IFS)

The proposed IFS scheme is an extension of the CW scheduling scheme of [5], hence it follows a high-to-low column-weight order for the variable nodes with unequal column weights. In addition, a new objective of the IFS scheme is to determine the scheduling order for the equal column-weight variable nodes, ensuring that each selected node receives maximum number of latest message updates from previously updated variable nodes. In other words, the variable nodes $\{v_1, v_2, \ldots v_N\}$ are divided into different groups according to their column-weight. Thereafter, following the high-to-low column-weight order, the individual groups are scheduled using the IFS scheme. To achieve this, we define two set of counters, namely the *check-counter* (ζ_m) for $1 \leq m \leq M$, and the *var-counter* (η_n) for $1 \leq n \leq N$. Given the iteration count i, the value of ζ_m represents the number of already updated variable nodes connected with check node c_m, whereas the quantity η_n represents the cumulative sum of ζ_m for $c_m \in \mathcal{M}(v_n)$, as given by

$$\eta_n = \sum_{c_m \in \mathcal{M}(v_n)} \zeta_m \tag{4}$$

In this perspective, the value of η_n reflects the total number of latest message updates accessible to variable node v_n. Thus, the IFS scheme finds the sequence of variable nodes on the basis of maximum value of η_n.

To begin with, all ζ_m and η_n are initialized to zero. This is to represent the starting point of a new decoding iteration. The IFS scheme proceeds to select the variable node associated with the largest η_n. In case of multiple variable nodes sharing the same maximum η_n, it randomly selects any one among them, say v_i, and records the scheduling order into a first-in-first-out (FIFO) structure $\boldsymbol{S}$. Then, the check-counter ζ_m for all $c_m \in \mathcal{M}(v_i)$ are incremented and the relevant var-counter are updated according to (4). This procedure is repeated until all the variable nodes are scheduled. The detailed steps of IFS scheme are outlined in Algorithm 1. Note that the IFS scheme needs to be run only once in the off-line mode, and the stored scheduling order is retrieved at the time of decoding, thereby precluding additional run-time computations.

Algorithm 1 Informed Fixed Scheduling (IFS).

```
Initialize η_n = 0, ∀1 ≤ n ≤ N ⇒ variable-counter;
Initialize ζ_m = 0, ∀1 ≤ m ≤ M ⇒ check-counter;
while S ∉ {v_1, v_2, ..., v_N} do
    Find i = arg max_{n={1,2,...,N}} {η_n};
    Select variable node v_i and store into vector S;
    Reset η_i = 0;
    for every c_m ∈ M(v_i) do
        Increment check-counter: ζ_m = ζ_m + 1;
        for every v_n ∈ N(c_m) \ v_i do
            Increment variable-counter: η_n = η_n + 1;
End
```

Example: To show the working of IFS scheme, we use a simple code graph, as shown in Fig. 1. Since this is an irregular code graph, column-weight 3 variable nodes $\{v_5, v_6, v_7\}$ are scheduled first followed by column-weight 2 variable nodes $\{v_1, v_2, v_3, v_4\}$. In the first place, variable node v_5 is selected as η_5, η_6 and η_7 are all equal to 0. Then, the check-counter ζ_1, ζ_4 and ζ_5 are incremented since c_1, c_4 and c_5 are connected with v_5, and the relevant var-counter values are updated, as shown in Fig. 1(b). Among the remaining two column-weight 3 variable nodes, v_7 is selected next on the account of larger η_7, and the respective check-counter and var-counter values are updated, as shown in Fig. 1(c). Similarly, the waiting column-weight 3 variable node v_6 is selected afterwards, as shown in Fig. 1(d). Likewise, the column-weight 2 variable nodes are scheduled, as shown in Fig. 1(f). The final sequence obtained in this example is given by $v_5 \Rightarrow v_7 \Rightarrow v_6 \Rightarrow v_3 \Rightarrow v_1 \Rightarrow v_2 \Rightarrow v_4$. In contrast, the conventional shuffled BP decoder will process variable nodes according to $v_1 \Rightarrow v_2 \Rightarrow v_3 \Rightarrow v_4 \Rightarrow v_5 \Rightarrow v_6 \Rightarrow v_7$,

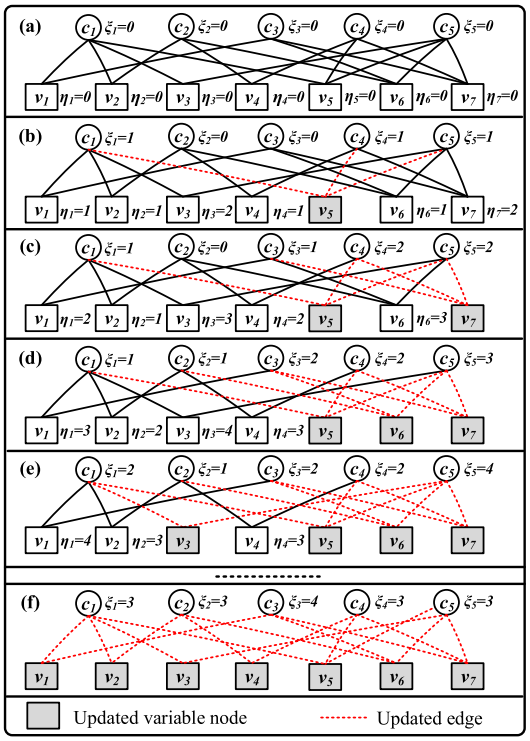

Fig. 1. Variable node scheduling $\{v_1, \ldots, v_7\}$ using the proposed *IFS* scheme: scheduled order $v_5 \Rightarrow v_7 \Rightarrow v_6 \Rightarrow v_3 \Rightarrow v_1 \Rightarrow v_2 \Rightarrow v_4$.

whereas the CW scheduling will follow the sequence $v_7 \Rightarrow v_6 \Rightarrow v_5 \Rightarrow v_4 \Rightarrow v_3 \Rightarrow v_2 \Rightarrow v_1$.

IV. IFS Performance Evaluation

A. Normalized Complexity and Error-Rate Performance

We perform Monte-Carlo simulations to plot the decoding convergence and error-rate performance for different regular and irregular LDPC codes. We construct four irregular quasi-cyclic (QC) LDPC codes based on IEEE WIMAX [6], IEEE WIFI [7], ETSI DVB-S2 [8] and CCSDS TM [9] standards with block-length 1152, 1944, 16200 and 20480 bits, and rate-1/2, rate-2/3, rate-11/15 and rate-4/5, respectively. Additionally, we also create three regular column-weight 3 LDPC codes with block-length 1000, 816 and 755 bits, and rate-1/2, rate-2/5 and rate-1/3, considering PEG [10], Gallanger's [11] and QC [12] code construction methods, respectively.

To show the decoding convergence, we set the maximum iteration number ($I_{\max}$) to 10. We execute the shuffled BP decoder employing both the conventional and IFS based scheduling schemes and record the average number of iterations required for error-rate convergence. We normalize the average number of iterations with respect to the conventional shuffled BP decoder. Specifically, the normalized average iterations represents the ratio $\frac{I_{\text{IFS}}}{I_{\text{conv}}}$, where I_{IFS} and I_{conv} denote the average number of iterations used by the IFS and conventional shuffled BP decoders, respectively. This normalization is performed to manifest the relative improvement in the convergence speed. For comparison, we also report the performance curves for the CW scheduling [5], given by the ratio $\frac{I_{\text{CW}}}{I_{\text{conv}}}$.

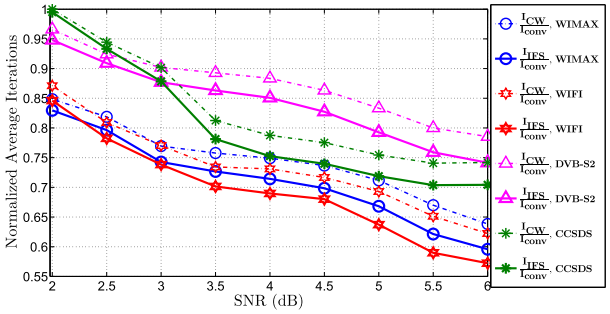

Fig. 2. Reduction in average iterations employing the proposed IFS and CW [5] schemes compared to the conventional shuffled scheme [2], simulated at $I_{\max} = 10$: using irregular LDPC codes [6]–[9].

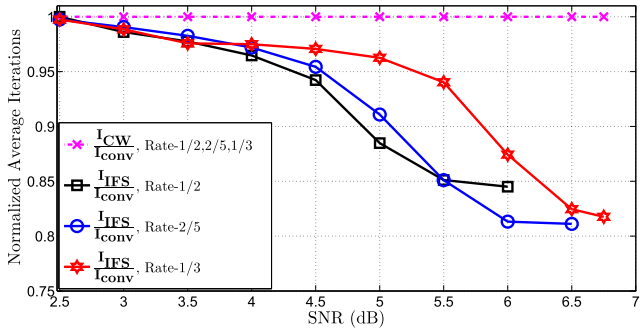

Fig. 3. Reduction in average iterations employing the proposed IFS and CW [5] schemes compared to the conventional shuffled scheme [2], simulated at $I_{\max} = 10$: using regular LDPC codes [10]–[12].

For irregular LDPC codes, the normalized convergence performance is shown in Fig. 2. Using the proposed IFS scheme, we notice an improvement by up to 45% at medium-to-high SNR. For the regular LDPC codes, faster convergence speed by up to 20% is realized, as shown in Fig. 3. To show the error-rate performance, we plot the frame-error-rate (FER) curves for the WIMAX [6] and WIFI [7] LDPC code under the BI-AWGN channel at $I_{\max} = 5$ and $I_{\max} = 10$, as shown in Fig. 4. It can be observed that the IFS scheme outperforms the conventional shuffled decoder at low iteration counts, and yet suffers no error-rate degradation at high iteration counts. Note that we have also performed simulations over Rayleigh fading channel and observed similar performance improvement.

B. Group-Shuffled BP Decoding

We further evaluate the performance of the proposed IFS scheme under the semi-serial group-shuffled BP decoding scheme for realizing a high throughput decoder implementation. The group-shuffled decoding scheme performs parallel sub-matrix processing in which, instead of single variable update, t variable nodes are simultaneously updated, where t is an arbitrary group size. To cater for group-shuffled decoding, the IFS algorithm should be modified according to the group size t, such that t maximum variable counters will be selected in Step 4 of Algorithm 1, and subsequently all the relevant variable and check counters will be incremented. In Fig. 5, we plot the normalized iterations for WIMAX [6] LDPC code for different values of t by employing the IFS scheme. Note that the curves in Fig. 5 are normalized w.r.t. the group-shuffled decoding by employing the conventional scheduling.

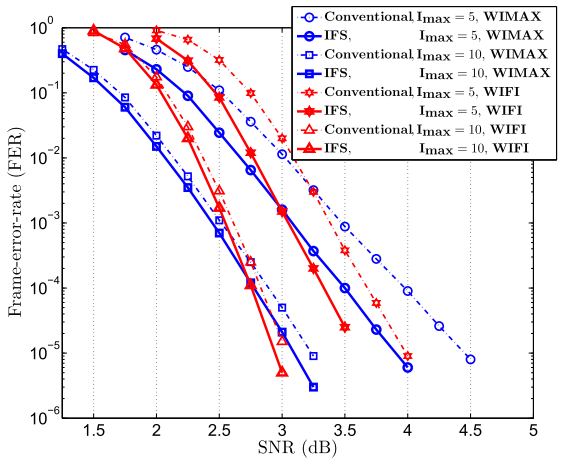

Fig. 4. FER performance employing the proposed IFS (*solid curves*) and conventional shuffled [2] (*dotted curves*) schemes, simulated at $I_{\max} = 5, 10$: using WIMAX [6] and WIFI [7] LDPC codes.

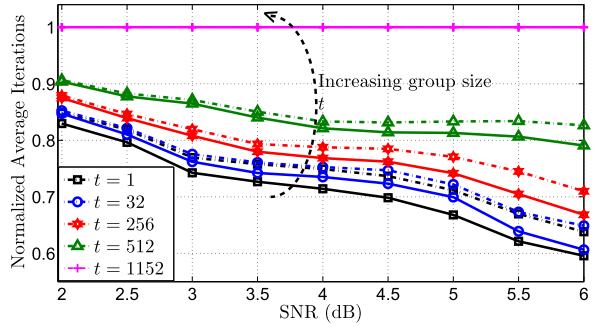

Fig. 5. Reduction in average iterations employing the proposed IFS (*solid curves*) and CW [5] (*dotted curves*) schemes compared to the conventional shuffled scheme [2], simulated under the group-shuffled decoding with group size t: using WIMAX [6] LDPC code.

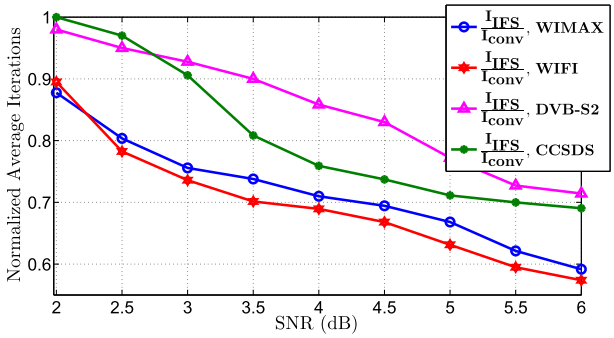

Fig. 6. Reduction in average iterations employing the proposed IFS scheme compared to the conventional shuffled scheme [2], simulated under the min-sum approximation with 8-bit fixed-point precision.

These results again show that the IFS scheme provides significant complexity reduction under the group-shuffled decoding, specifically for smaller values of t. It is pertinent to mention that the variable node grouping can also be done based on the code structure, instead of random grouping with arbitrary size t, and similar performance improvement can be realized.

C. Fixed-Point Precision With Min-Sum Approximation

For hardware implementation of BP decoder, the LLR values, and subsequently the reliability updates, are converted into fixed-point data type with b bits of precision. Apart from that, in most real world implementations, the sum-product BP decoder is replaced by a low-complexity *min-sum* BP decoder [13] at the expense of slight error-rate degradation. Using 8-bit fixed-point precision, with 5 bits for the integer part and 3 bits for the fractional part, we perform the min-sum shuffled BP decoding for irregular LDPC codes [6]–[9], and plot the normalized complexity in Fig. 6. This figure shows that the proposed IFS scheme yields similar performance improvement under the min-sum decoding with fixed-point simulation.

V. Conclusion

An improved scheduling scheme for the belief-propagation (BP) decoding of LDPC code is presented. In this scheme, a fixed sequence of variable nodes is determined based on the maximum number of newly available message updates. Since the variable nodes are scheduled only once in the off-line mode, the proposed scheduling scheme does not incur additional run-time computations. This sequence of variable nodes is then serially (or semi-serially) decoded under the sum-product or min-sum shuffled BP algorithms with floating-point or fixed-point precision. Simulation results show a significant improvement in the decoding convergence speed in the range of medium-to-high signal-to-noise ratio (SNR). Moreover, the error performance improves for the initial decoding iterations and remains comparable to the conventional shuffled BP decoding scheme for subsequent decoding iterations.

References

[1] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1963.
[2] J. Zhang and M. P. C. Fossorier, "Shuffled iterative decoding," *IEEE Trans. Commun.*, vol. 53, no. 2, pp. 209–213, Feb. 2005.
[3] D. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. IEEE Workshop. Signal Process. Syst.*, Oct. 2004, pp. 107–112.
[4] H.-C. Lee and Y.-L. Ueng, "LDPC decoding scheduling for faster convergence and lower error floor," *IEEE Trans. Commun.*, vol. 62, no. 9, pp. 3104–3113, Sep. 2014.
[5] C. A. Aslam, Y. L. Guan, and K. Cai, "Improving the belief-propagation convergence of irregular LDPC codes using column-weight based scheduling," *IEEE Commun. Lett.*, vol. 19, no. 8, pp. 1283–1286, Aug. 2015.
[6] *LDPC Coding for OFDMA PHY*, IEEE Standard C802.16e-05/066r3, Aug. 2005.
[7] *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11, 2012.
[8] *DVB-S2 Adaptive Coding and Modulation for Broadband Hybrid Satellite Dialup Applications*, document ETSI TS 102 441, 2005.
[9] *TM Synchronization and Channel Coding*, document CCSDS 131.0-B-2, 2011.
[10] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.
[11] D. J. C. Mackay. *Encyclopedia of Sparse Graph Codes*. Accessed on Oct. 1, 2016. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html
[12] R. Tanner *et al.*, "LDPC block and convolutional codes based on circulant matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 12, pp. 2966–2984, Dec. 2004.
[13] M. P. C. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999.