ITERATIVE MESSAGE PASSING TECHNIQUES FOR RAPID CODE ACQUISITION

Mingrui Zhu and Keith M. Chugg
Communication Science Institute
Dept. of Electrical Engineering
University of Southern California, Los Angeles 90089-2565
{mingruiz,chugg}@usc.edu

***Abstract* —Iterative message passing algorithms (MPAs) have found application in a wide range of data detection problems because they can provide near optimal performance and significant complexity reduction. In this paper, we demonstrate that they can be used to efficiently solve the pseudo random code acquisition problem as well. To do this, we represent good pseudo-noise (PN) patterns using sparse graphical models, then apply the standard iterative message passing algorithm over this graph to approximate maximum likelihood synchronization. Simulation results show that this algorithm achieves better performance than traditional serial search code acquisition in the sense that it works at low signal-to-noise ratios (SNRs) and is much faster. Compared to full parallel search, this approach typically provides significant complexity reduction.**

I. INTRODUCTION

ACHIEVING and maintaining code synchronization between the transmitter and the receiver is critical to all spread spectrum systems because even a small misalignment can cause serious signal-to-noise ratio (SNR) degradation. Typically, this task is performed in two steps: code acquisition and code tracking. Our focus here is to apply iterative message passing algorithms (MPAs), generalizations of the famous Turbo Decoding Algorithm[3], to the acquisition of m-sequences, wherein the ultimate goal is to bring two codes into coarse time alignment (*i.e.,*within one code-chip interval).

For years, there have been two major acquisition strategies: parallel search and serial search. The former is generally useful for theoretical analysis only because, while it provides Maximum Likelihood (ML) detection of the initial state, it is usually too complex to be implemented. The latter has good performance, in the sense of probability of acquisition, even for low SNR, but it is slow.

In some applications, it may be desirable to use very long pseudo-noise (PN) patterns and still acquire the sequence epoch based on relatively few observations (*i.e.,*a small fraction of the PN period). Ultra-Wideband Radio (UWB) systems provide a good example. A sample waveform for a low-duty cycle UWB signal is shown in Figure 1(a), where the time between wideband pulses is referred to as the frame time. The PN acquisition problem is also diagrammed in Figure 1(a) in terms of a search over potential bins. Note that this problem is more difficult than the corresponding classical spread-spectrum PN acquisition problem for two reasons. First, the frame epoch must be acquired simultaneously with the PN pattern. This implies that for each resolution bin on the frame epoch, there is a complete PN search space (*i.e.,*"PN phase wheel"). Second, for a given hypothesized frame epoch, there are typically more resolution bins to search in the UWB case because the larger bandwidths employed imply finer time resolution. The direct sequence spread-spectrum (DS/SS) case is shown in Figure 1(b). Note that the acquisition uncertainty is only in the PN code phase because the DS waveform has a 100% duty cycle.

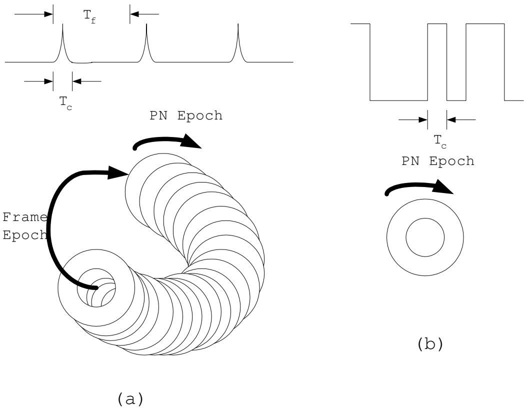

Fig. 1. A sample waveform and diagram of the associated PN acquisition problem for two spread spectrum systems. (a) a low duty cycle UWB system the frame epoch and PN code phase must be determined and (b) a direct sequence system where on PN code phase need be acquired. Note that the DS system is modelled as its complex baseband equivalent signal.

If long PN patterns are to be used in covert UWB systems, extremely fast PN acquisition is required. This is not only due to the high level of timing uncertainty as described above, but also the fact that the true frame epoch will certainly drift due to oscillator imperfections and/or platform mobility. More specifically, if the bins in Figure 1(a) were tested sequentially and the frame epoch was drifting, it is possible that the search will never locate the true epoch – *i.e.,*this may result in a "chasing one's tail" situation.

This UWB example provides the motivation for the proposed approach. Specifically, it would be desirable to search all possible PN pattern phases for a fixed, hypothesized frame epoch in parallel. It is also desirable to complete this search based on a relatively small number of observations and with reasonable implementation complexity. The method presented in this paper provides an attractive solution to this problem that cannot be achieved using traditional PN acquisition strategies.

We describe the m-sequences and associated graphical representations in section II. Section III contains a short review of traditional methods. Section IV defines the message passing algorithm. Simulation results are provided in section V. Section VI contains conclusions and suggestions for future work.

This work was supported in part by the National Science Foundation CCR-0082987 and the Army Research Office DAAD19-01-1-0477

II. M-SEQUENCES AND GRAPHICAL REPRESENTATIONS

A. M-sequences

Linear feedback shift register (LFSR) sequences having the maximum possible period for an r-stage shift register are called maximal-length sequences or m-sequences[6]. They have been successfully employed in a wide range of spread spectrum systems and many other spreading codes can be derived from them. If the LFSR sequence is binary, the maximum achievable period of is $2^r - 1$. To achieve this maximum value, two conditions must be satisfied. First, the generating polynomial, which is defined as:

$$g(D) = g_0 + g_1 D + g_2 D^2 + ... + g_{r-1} D^{r-1} + g_r D^r \quad (1)$$

where D is the unit delay operator, is primitive. Second, the shift-register is loaded with non-zero initial state. The (infinitely long) periodic sequence $\underline{x}$ generated then can be written as

$$\underline{x} = ..., x_{-2}, x_{-1}, x_0, x_1, x_2, ...x_{N-1}, x_N, ...$$

where $N = 2^r - 1$ is the period of the m-sequence so that $x_{N+i} = x_i$. The goal of code acquisition is to find the phase of the sequence present in the received signal, where $\underline{x}$, $D\underline{x}$, $D^2\underline{x}$, ..., $D^{N-1}\underline{x}$ are defined as phases of $\underline{x}$.

In most practical scenarios, we can only observe part of this long sequence to acquire the initial phase, so the problem can be stated as: for a given number of L observations, $\{z_k\}_0^{L-1}$, estimate the initial state $\{u_i\}_0^{r-1}$. Usually, $r \ll L \ll N$. If no data modulation is present,

$$z_k = \sqrt{E_c}(-1)^{x_k} e^{-j\theta_c} + n_k, 0 \leq k \leq L-1 \quad (2)$$

where n_k is white gaussian noise with one-sided power spectrum density of N_0 and θ_c is the carrier phase. This general model can be applied to both to UWB and DS/SS systems. In a UWB system, as there is no carrier, $\theta_c = 0$. In DS/SS system, if the phase of the RF carrier is known, $\theta_c = 0$ as well. However, in practice, θ_c is typically unknown at the point of PN acquisition because the SNR before despreading is too low to enable carrier synchronization. This implies non-coherent PN acquisition. In this case, θ_c is modelled as random variable uniformly distributed over $[0, 2\pi]$ which is constant over the period of L observations.

An r-stage LFSR is shown in Figure 2. At any given time k, let $S_k^{(i)}$, $0 \leq i \leq r-1$, be the value of the i^{th} register and x_k be the output, the following feedback equation is satisfied.

$$\begin{aligned} 0 &= g_0 S_k^{(r)} \oplus g_1 S_k^{(r-1)} \oplus ... \oplus g_{r-1} S_k^{(1)} \oplus g_r S_k^{(0)} && (3) \\ x_k &= S_k^{(0)}, 0 \leq k \leq L-1 && (4) \end{aligned}$$

where $\oplus$ is modulo 2 addition.

As $S_k^{(i)} = S_{k+1}^{(i-1)}$, $1 \leq i \leq r$, and $g_0 = g_r = 1$ when $g(D)$ is primitive,

$$S_k^{(r)} = S_k^{(0)} \oplus g_1 S_k^{(r-1)} \oplus ... \oplus g_{r-1} S_k^{(1)} \quad (5)$$

Define $\mathbf{S}_k = (S_k^{(0)}, S_k^{(1)}, ..., S_k^{(r-1)})$ as the state, and let the initial state $\mathbf{S}_0$ be $\mathbf{u} = (u_0, u_1, ..., u_{r-1})$, then (4) and (5) together define the transition $t_k = (\mathbf{S}_k, x_k, \mathbf{S}_{k+1})$. This is a finite state machine, whose evolution is determined entirely by its initial state.

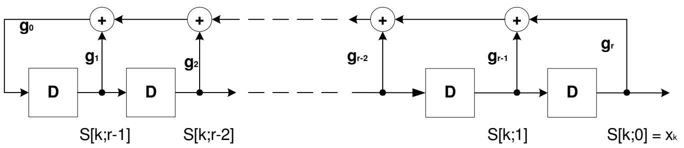

Fig. 2. An r-stage LFSR

B. Graphical representations of m-sequence

Graphical models are powerful tools to formalize joint probability distribution problems as they can represent those problems compactly and intuitively, and can yield efficient algorithms. The trellis representation along with the well known Viterbi algorithm is an excellent example.

There are a number of conventions for representing system graphically [1], [2], [7], [11], [12]. Given a graphical model or block diagram, there is standard message passing algorithm [1], [2], [4], [8]. The most important aspect of the graphical model is whether it contains undirected cycles. If cycles do not exist, the standard MPA will find the optimal solution (*i.e.,*ML detection) given a loose requirement on the message passing schedule. If cycles do exist, one may still apply the standard MPA, but in this case optimality is not assured. This results in an iterative MPA (iMPA), which is an effective heuristic. For many problems, an iMPA resulting from a loopy graph will be significantly less complex than the optimal MPA resulting from a cycle-free graphical model. Thus, the value of the iMPA heuristic is that it can provide a good (not optimal) solution with relatively low complexity. The standard turbo decoding algorithm is an example – *i.e.,*the iterative decoder is an iMPA that results in near-optimal performance at a fraction of the complexity required for ML decoding.

Extracting a graphical model from an input-output relationship is a difficult problem. Some approaches exist for cycle-free representations [1], [2], [5], however, extracting loopy graphical models resulting in effective iMPAs remains largely an art. Specifically, the art is to identify the proper local structure and establish sparse connections between these local structures. Equivalently, one must identify appropriate hidden variables (*i.e.,*the art), after which a number of conventions can be used to establish the loopy graphical model [1], [2], [7], [11], [12].

Consider an example related to the problem at hand. Given the generating polynomial for an 15-stage LFSR as [1400001], i.e., $g(D) = D^{15} + D^1 + D^0$, then,

$$0 = S_k^{(15)} \oplus S_k^{(14)} \oplus S_k^{(0)} \Longleftrightarrow x_k = x_{k-1} \oplus x_{k-15} \quad (6)$$

One way to represent this system is to label **state vertices**, **transition vertices** and **output vertices** by sets $\{\mathbf{S}_k\}$, $\{t_k\}$ and x_k, respectively, and connect any pair of vertices by an edge if the intersection of those two vertices is non-empty. A cycle-free graphical representation is shown in Figure 3. Running the optimal MPA on this cycle-free graph is complex, with the standard parallel search strategy viewed as a simplification to this approach. We call Figure 3 “full-state representation”.

A loopy representation can be found by defining $\mathbf{S}_k^*$ as (x_{k-1}), $\mathbf{t}_k^*$ as $(\mathbf{S}_k^*, x_{k-15}, x_k, \mathbf{S}_{k+1}^*)$, and then use them to label state vertices and transition vertices and connect $\mathbf{t}_k^*$ with $\mathbf{S}_k^*$,

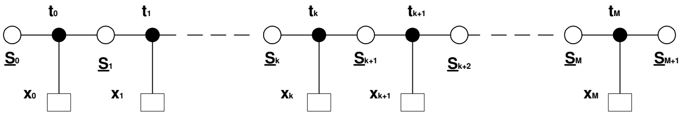

Fig. 3. Cycle-free graphical representation of an r-stage LFSR, *i.e.,*full-state representation

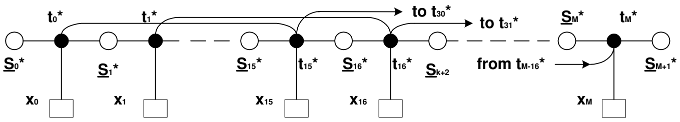

Fig. 4. Sparse Loopy Representation of the 15-stage shift-register with $g(D) = D^{15} + D^1 + D^0$.

$\mathbf{S}^*_{k+1}$, x_k and $\mathbf{t}^*_{k+15}$. This is a "sparse loopy representation" and is shown in Figure 4.

Another possible, but not so straightforward, way to find the loopy representation is to define $\mathbf{S}^*_k$ as (x_{k-1}, x_{k-2}) instead. This graph has the same topology as shown in Figure 4, but with different vertex labels. Though x_{k-2} does not appear in equation (6) so that it contributes nothing explicitly, introducing it improves the performance and speeds up the rate of convergence of iterative MPAs running over sparse loopy graphs. It is also possible to introduce more variables into the definition of $\mathbf{S}^*_k$. However, the complexity of the iterative MPAs grows exponentially to the number of bits of $\mathbf{S}^*_k$ and no significant improvement of the performance can be observed.

III. Traditional Acquisition Algorithms

As uniform a priori probability is assumed for the initial state, both full parallel search and serial search calculate $p(\mathbf{z}|\mathbf{u})$, the likelihood of $\mathbf{u}$, where $\mathbf{u}$=$(u_0(0), u_1(0), ..., u_{r-1}(0))$ and $\mathbf{z}$= $(z_0, z_1, ..., z_{L-1})$. In the former, $p(\mathbf{z}|\mathbf{u})$ for all candidates is calculated, and the estimated initial state is $\hat{\mathbf{u}} = \arg\max_{\mathbf{u}} p(\mathbf{z}|\mathbf{u})$ This provides ML estimation of initial state. However, since full parallel search has memory requirements exponential to the length of the LFSR, it is often impractical.

Though $p(\mathbf{z}|\mathbf{u})$ is calculated in serial search as well, the whole set of possible states is not necessarily searched. Instead, a threshold is set and the whole set of candidate states is divided into a number of cells. Cells are examined serially by correlating the corresponding local sequence with the channel observations over the "dwell time", τ_d. If a correlation larger than the threshold is observed, a successful "acquisition" is declared. Otherwise, current observations are discarded, correlation over another τ_d is computed to test another cell. This process is continued until acquisition is declared.

Though this approach can significantly reduce the memory requirements, and works well at low SNRs, it is slow. Without a priori information on the phase, the mean acquisition time for the simple single-dwell serial search [10] is:

$$\bar{T}_{ACQ} = [\frac{2 + (2 - P_D)(q - 1)(1 + KP_{FA})}{2P_D}] \cdot \tau_d \qquad (7)$$

where P_D is the probability of detection for a single-dwell test, P_{FA} is the probability of false alarm, K is the penalty time for a false alarm, and q is the total number of cells. Consider the best case, where $P_D = 1$ and $P_{FA} = 0$, that is, whenever the right cell is reached, acquisition is declared and declaring an acquisition at the wrong cell never happens. Also assuming the total number of cells is the total number of possible states, then $q = 2^r - 1$ and $\frac{\bar{T}_{ACQ}}{\tau_d} = (q + 1)/2 = 2^{r-1}$. If both the observation length in parallel search and $\frac{\tau_d}{T_c}$ of serial search are L, only L observations are needed for parallel search and it can acquire the initial state in time LT_c, but on average at least $\frac{(q+1)L}{2} = 2^{r-1}L$ observations are needed for serial search and it takes $\frac{(q+1)LT_c}{2} = 2^{r-1}LT_c$ to acquire.

Degree	Octal representation of generating polynomial
15	[100003],[140001],[100021],[104001]
17	[400011],[440001],[400041],[410001]
18	[1000201],[1004001],[1000077],[1760001]
29	[4000000005],[5000000001]
31	[20,000,000,011],[22,000,000,001]

TABLE I

Examples of sparse generating polynomials for m-sequences.

IV. Iterative Message Passing Algorithms

Message passing algorithms have been studied intensely since the invention of Turbo Codes[3]. It is well known that once the graph is given, the standard processing is well defined and only schedule needs to be specified. The standard message passing algorithms referred to in this paper are well defined in [1], [2], [4], [7], [8], [11], [12], and iterative message passing algorithms or iMPAs refer to MPAs on loopy graphs.

To apply message passing techniques to solve joint probability distribution problems, we usually start by formalizing the problems using graphical models, defining the messages passed along edges and specifying the schedule. If the graphical representation is not loopy, optimality can be achieved. The full-state graphical representation is acyclic, (*i.e.,*it is a tree), so maximum likelihood decisions are obtained by running a standard message passing algorithm on it. On the other hand, there are many cycles of some fixed length in the sparse loopy one, such as Figure 4, optimal decisions may not be achieved using iMPAs. However, the complexity of the MPA is a function of the underlying graph. The former has complexity exponential in r, the shift register memory, whereas the latter has complexity exponential to the number non-zero coefficients of $g(D)$.

Our approach tries to approximate the full parallel search when the generating polynomial $g(D)$ is sparse, i.e., there are only a few, compared to r, isolated $1's$. An interesting observation for m-sequences of long period is that, many have sparse generating polynomials which directly provide sparse loopy representations. In Table I, some examples are given[9]. In Figure 4, one sparse loopy representation for [1400001] of degree 15 is presented.

Messages passed over Figure 4 are obtained following the standard rules [1], [2], [4], [7], [8], [11], [12]. Specific expressions of the messages that are passed in this scheme in terms of the channel observations are given in the Appendix.

There are many different schedules for MPAs on acyclic graphs that all converge to the same (optimal) solution [1], [2]. However, scheduling can significantly affect the performance of MPAs on loopy graphs. Furthermore, when MPAs are running on loopy graphs, convergence to a local optimum is not guaranteed, but is usually observed empirically.

Referring to Figure 4, the schedule used for the iMPA in this paper is as follows. First, messages are sent along the edges connecting $\mathbf{t}_k^*$ and $\mathbf{t}_{k+15}^*$. Then, messages are passed forward along the edges connecting $\mathbf{S}_k^*$, $\mathbf{t}_k^*$, and $\mathbf{S}_{k+1}^*$, sequentially staring from $k = 0$. Next, messages are passed backward along the edges connecting $\mathbf{S}_{k+1}^*$, $\mathbf{t}_k^*$, and $\mathbf{S}_k^*$, sequentially staring from $k = L$. This defines one iteration. The forward and backward processing can be viewed as running the standard forward-backward algorithm (FBA) on the trellis defined in the absence of the edges from $\mathbf{t}_k^*$ and $\mathbf{t}_{k+15}^*$. Thus, the algorithm may be viewed as alternatively running the FBA and passing messages over the loops defined by the edges connecting $\mathbf{t}_k^*$ and $\mathbf{t}_{k+15}^*$. Each successive run of the FBA differs because the messages passed over the loops alter the effective state transition metrics.

After running iMPA over the loopy graph for a number of iterations, the iMPA is stopped and soft-output information of the first 15 bits is calculated and thresholded to obtain a hard decision for the initial state. The appendix contains brief short pseudo-code for this iMPA.

V. Simulation results

Computer simulations are done for both UWB (pulsed) systems, where there is no carrier, and traditional DS system, where the effect of unknown carrier phase must be considered. In both cases, we use the m-sequence generated by an 15-stage shift register with generating polynomial of [1400001].

A. *UWB (pulsed) Systems*

A.1 Evaluating the algorithm

We evaluate our algorithm using the probability of acquisition v.s. chip SNR, E_c/N_0, the acquisition time and the complexity.

The acquisition probabilities of different acquisition schemes are plotted in Figure 5. It can be seen that the 4-state *min*-sum* iMPA with is about 2.2 *dB* worse than the ML exhaustive search due to the regular structure and short fixed-length cycles in the graphical representation, and 0.3 *dB* better than the simple serial search. The 4-state *min-sum* iMPA has the performance very close to the simple serial search.

The acquisition times of these algorithms are shown in Figure 5 as well. Both full parallel search and iterative MPAs can achieve code acquisition with time duration comparable to $128T_c$. On the other hand, the mean acquisition time of simple serial search is $2.09 \cdot 10^6 T_c$. Thus, the iterative MPAs are 16000 times faster than the simple serial search. Since the penalty time was assumed to be zero when $\overline{T}_{ACQ}$ is calculated, 16000 is a conservative estimate.

To compare the complexity of these algorithms, both memory requirements and the total number of arithmetic operations, denoted by R_m and R_a respectively, are considered. Those numbers, along with acquisition time, are summarized in Table II. Values in parenthesis correspond to numerical results obtained using $L = 128$, $N_c = 100$ and $r = 15$.

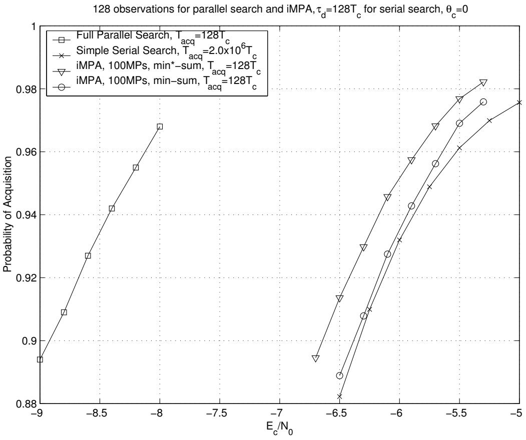

Fig. 5. Probability of acquisition (from left to the right): full parallel search, 100-iteration *min*-sum* iMPA, 100-iteration *min-sum* iMPA, where $\mathbf{S}_k^* = (x_{k-1}, x_{k-2})$, and simple serial search. The number of observations of both parallel search and the iterative detectors is 128, and the dwell time of serial search is 128 chip intervals.

	Parallel	Serial	iMPA
T_{ACQ}	LT_c $(128T_c)$	$2^{r-1}T_c$ $(2.09 \cdot 10^6 T_c)$	LT_c $(128T_c)$
R_m	$2^r(32736)$	1	$10L(1280)$
R_a	$2^r L$ $(4.19 \cdot 10^6)$	$2^r L$ $(4.19 \cdot 10^6)$	$32LN_c$ $(4.10 \cdot 10^5)$

TABLE II

T_{ACQ}, R_m AND R_a OF FULL PARALLEL SEARCH, SIMPLE SERIAL SEARCH AND PROPOSED ITERATIVE MPA. BOTH FULL PARALLEL SEARCH AND IMPA HAVE L OBSERVATIONS, $\tau_d = LT_c$ FOR SIMPLE SERIAL SEARCH, AND N_c IS THE NUMBER OF ITERATIONS THAT IMPA RUNS.

To evaluate the iMPA, we would also like to know the performance gain when increasing the number of iterations. For the *min-sum* iMPA with $L = 128$, when N_c is increased from 100 to 200, 0.3 *dB* is gained. There is another 0.2 *dB* gain achieved by increasing the number further to 400. No significant iteration gain is achieved after that.

As illustrated in Figure 6, doubling the length of the observation (L) provides approximately 3 *dB* improvement for the traditional serial and parallel search approaches. This is to be expected since doubling the number of observations roughly doubles the ratio between the partial period correlation under the correct (in-phase) and out-of-phase alignments. However, the performance of the iterative MPA is limited by the cycles, which are of the length of 15 in this example. Thus, direct application of this method is most attractive for short observation intervals, as motivated in section I. However, this iMPA could potentially be used sequentially or in parallel over multiple time windows of size 128 in order to capitalize on the additional observations.

A.2 Improving the performance

The performance of the proposed iMPAs over Figure 4 can be improved. One way to do that is to use a *soft information filter* [4], which simply filters out large variations in the soft information of a particular quantity from one iteration to the

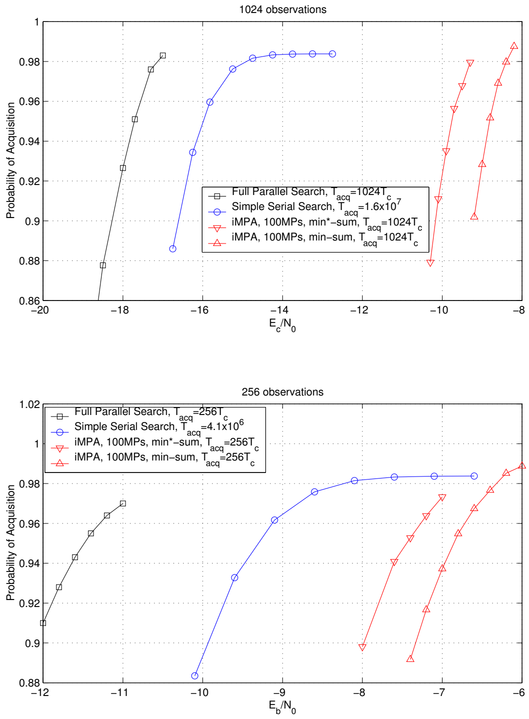

Fig. 6. The performance of full parallel and serial search, and iMPA when L is 1024 and 256, $\theta_c = 0$

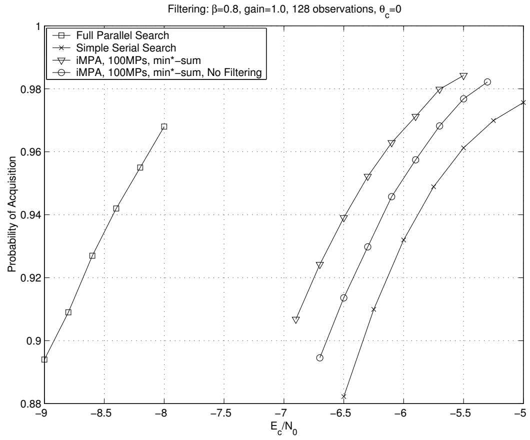

Fig. 7. Filtering: $\beta = 0.8$, $g = 1.0$, $\theta_c = 0$, $L = 128$

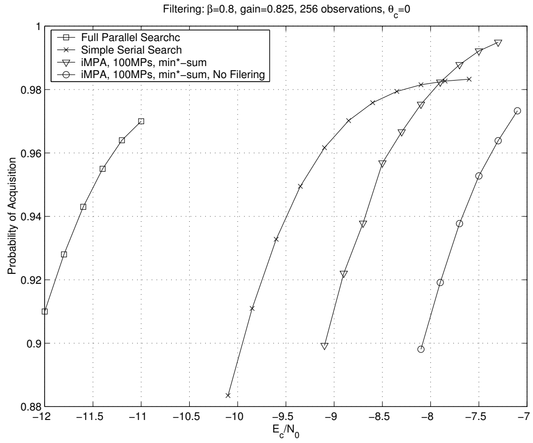

Fig. 8. Filtering: $\beta = 0.8$, $g = 0.825$, $\theta_c = 0$, $L = 256$

next. In this case, let the label of the edge connecting $\mathbf{t}_k^*$ and $\mathbf{t}_{k+15}^*$ be a_k, the standard extrinsic information after n iterations $SO^{(n)}[a_k]$ is filtered by a low-pass filter $g \cdot H(z)$ where $H(z) = h_1 + h_2 z^{-1} = \frac{\beta}{\sqrt{\beta^2+(1-\beta)^2}} + \frac{1-\beta}{\sqrt{\beta^2+(1-\beta)^2}} \cdot z^{-1}$ is a unit-gain low-pass filter, and g is the gain. The actual messages that are passed along edges are:

$$SO_f^{(n)}[a_k] = g \cdot (h_1 \cdot SO^{(n)}[a_k] + h_2 \cdot SO_f^{(n-1)}[a_k]) \qquad (8)$$

where $SO_f^{(n)}[a_k]$ is the filtered soft-out information. The parameter β is used to adjust the bandwidth of the filter. Specifically, when $\beta = 1$, there is no filtering.

It can be observed from Figure 7 that 0.3 dB is gained by filtering when $L = 128$. This number is 1.0 dB when $L = 256$. Filters with different β and different gain g were simulated. For $\beta = 0.8$ and $L = 128$, $g = 1.0$ is the empirically optimal scaling.

B. Traditional DS/SS systems

Since reliable estimation of the carrier phase is not possible prior to PN acquisition, a practical algorithm must operate without phase knowledge. It is assumed that there exists an unknown deterministic phase which changes slowly so that it is constant over the duration of L observations. Our approach is based on generalized likelihood [4] where only a finite number of candidate θ_c-value are considered. Specifically, 8 candidate phase values are considered, the MPA is run for each separately and the best epoch of these 8 phases is selected.

Simulation results of the 100-iteration MPA for this approach are shown in Figure 9 along with the curve of the ideal case where θ_c is known. The 8-phase quantization approach works well, at the cost of an increasing in complexity by a factor of 8, whereas an additional 2 dB degradation is observed for 4-phase quantization.

VI. Conclusion and Future Work

Iterative techniques are well known to be applicable in a wide range of applications, and in this paper we applied this principle to solve the code acquisition problem. Simulation results showed that the iterative message passing algorithm on the sparse graph worked by itself, for it can achieve acquisition at low SNRs and it works much faster than serial search. This approach is especially favorable when the block size is relatively small.

In practice, hybrid search is usually used as a compromise of full parallel search and simple serial search. However, hybrid search compromises linearly between memory requirement and acquisition time and it does not necessarily reduce the number of arithmetic operations, R_a, needed for an successful acquisition. Since the iterative message passing algorithm over a sparse graphical model can approximate full exhaustive search with significant reduction on both R_m and R_a, it is a promising approach for code acquisition problem.

"Data blind acquisition", i.e., to acquire the initial state of the PN code while data modulation is presented, designing spread-

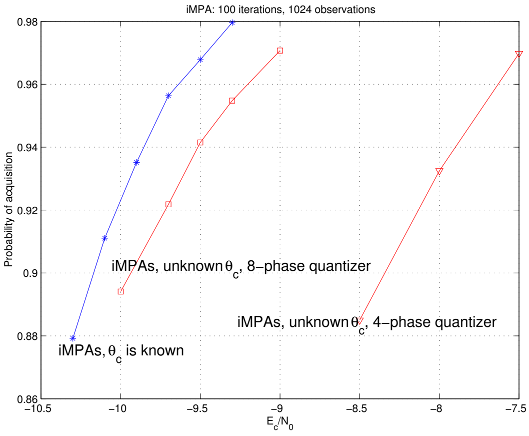

Fig. 9. Performance of iterative MPA in traditional DS/SS system,$L = 1024$

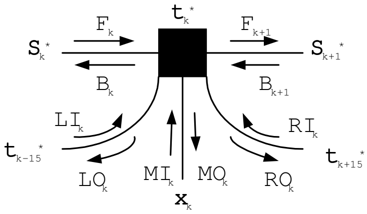

Fig. 10. Messages go into and go out from transition vertex t_k^*

ing codes with "good" autocorrelation property (not necessarily two value) and sparse generating polynomials are some of the interesting future research problems.

Appendix

To help clarify the messages passed along edges in Figure 4, transition vertex $\mathbf{t}_k^*$ and all connected edges are re-drawn in Figure 10. Also, specific labels are given to messages passed along different edges, as shown in Figure 10.

Define soft-in information in terms of channel observations as:

$$MI_k[x_k] = \frac{2\sqrt{E_c}z_k(-1)^{x_k}}{N_0} \tag{9}$$

Also, let

$$\begin{aligned} M_0^{L-1}[t_k^*] = \quad & \min_{t_k^*}(F_k[\mathbf{S}_k^*] + MI_k[x_k] + RI_k[x_k] \\ & + LI_k[x_{k-15}] + B_{k+1}[\mathbf{S}_{k+1}^*]) \end{aligned} \tag{10}$$

Then, the *min-sum* messages passed in this iMPA are:

$$MO_k[x_k] = min_{t_k^*:x_k} M_0^{L-1}[t_k^*] - MI_k[x_k] \tag{11}$$

$$F_{k+1}[\mathbf{S}_{k+1}^*] = min_{t_k^*:S_{k+1}^*} M_0^{L-1}[t_k^*] - B_{k+1}[\mathbf{S}_{k+1}^*] \tag{12}$$

$$B_k[\mathbf{S}_k^*] = min_{t_k^*:S_k^*} M_0^{L-1}[t_k^*] - F_k[\mathbf{S}_k^*] \tag{13}$$

$$LO_k[x_{k-15}] = min_{t_k^*:x_{k-15}^*} M_0^{L-1}[t_k^*] - LI_k[x_{k-15}] \tag{14}$$

$$RO_k[x_k] = min_{t_k^*:x_k^*} M_0^{L-1}[t_k^*] - RI_k[x_k] \tag{15}$$

$$RI_k[x_k] = LO_{k+15}[x_k] \tag{16}$$

$$LI_k[x_{k-15}] = RO_{k-15}[x_{k-15}] \tag{17}$$

where the notation $u : v$ means all u consistent with v, $x_k \in \{0, 1\}$ and $\mathbf{S}_k^* \in \{00, 01, 10, 11\}$. Similarly, *min*-sum* messages can be obtained by replacing *min* operators in equations (10)-(15) by *min**, where $min*(x, y) = min(x, y) - \ln(1 + e^{-|x-y|})$.

Pseudo-code of the proposed iterative message passing algorithm

1. Initialization.

$$MI[x_k] \leftarrow \frac{2\sqrt{E_c}z_k(-1)^{x_k}}{N_0},$$

$$LI_k[x_{k-15}] \leftarrow MI[x_{k-15}], RI_k[x_k] \leftarrow MI[x_{k+15}]$$

$$F_0[\mathbf{S}_0^*], B_L[\mathbf{S}_L^*], N_{iteration} \leftarrow 0$$

2. Forward-backward algorithm: updating $F_k[\mathbf{S}_k^*]$ and $B_k[\mathbf{S}_k^*]$, $0 \le k \le L - 1$, sequentially using (12) and (13), respectively.

$$F_0[\mathbf{S}_0^*] \rightarrow F_1[\mathbf{S}_1^*] \rightarrow ...F_k[\mathbf{S}_k^*]... \rightarrow F_L[\mathbf{S}_L^*]$$

$$B_L[\mathbf{S}_L^*] \rightarrow ...B_{k+1}[\mathbf{S}_{k+1}^*] \rightarrow B_k[\mathbf{S}_k^*]... \rightarrow B_0[\mathbf{S}_0^*]$$

3. Updating $LO_k[x_{k-15}]$ and $RO_k[x_k]$, $0 \le k \le L - 1$, using equation (14) and (15) respectively. Then,

$$N_{iteration} \longleftarrow N_{iteration} + 1$$

$$LI_k[x_{k-15}] \longleftarrow RO_{k-15}[x_{k-15}], RI_k[x_k] \longleftarrow LO_{k+15}[x_k]$$

4. If $N_{iteration}$ is less than R_a, go back to step 2; otherwise, estimating the initial state using $M[u_i] = M[x_i] = MI_i[x_i] + MO_i[x_i]$, $0 \le i \le 14$. The decision rule is: $u_i = 0$ when $M[u_i] \ge 0$, and $u_i = 1$ otherwise.

References

[1] S. M. Aji, "Graphical Models and Iterative Decoding," *Ph.D Thesis, CalTech*, 1999

[2] S. M. Aji and R. J. McEliece, "The Generalized Distributive Law", *IEEE Trans. on Info. Theory*, vol. 46, No. 2, pp. 325-343, Mar, 2000.

[3] C. Berrou and A. Glavieux and P. Thitmajshima, "Near Shannon limit error-correcting coding and decoding: turbo-codes", *ICC, Geneva, Switzerland*, 1064-1070, May, 1993

[4] K. M. Chugg, A. Anastasopoulos, and X. Chen,*Iterative Detection: Adaptivity, Complexity Reduction, and Applications*, Kluwer Academic Publishers, MA, 2001.

[5] T. H. Cormen and C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, mit, Cambridge, Mass, 1990.

[6] Solomon W. Golomb, *Shift Register Sequences, revised edition*, Aegean Park Press, 1982.

[7] F. R. Kschischang and B. J. Frey and H.-A. Loeliger, "Factor graphs and the sum-product algorithm", *IEEE Trans. on Info. Theory*, 2000.

[8] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1988.

[9] Roger L. Peterson, Rodger E. Ziemer, David E.Borth,*Introduction to Spread-Spectrum Communications*, Prentice Hall, NJ, 1995.

[10] A. Polydoros, C. L. Weber, "A unified approach to serial search spread-spectrum code acquisition," *IEEE Trans. on Comm.*, vol. 32, No. 5, pp. 542-560, May, 1984

[11] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Information Theory*, vol. 27, pp. 533-547, Sep., 1981

[12] N. Wiberg, "Codes and Decoding on General Graphs," *Ph.D Thesis, Linköping University (Sweden)*, 1996