A Data Modem for GSM Voice Channel

Christoph K. LaDue, Vitaliy V. Sapozhnykov, and Kurt S. Fienberg

***Abstract*—This paper introduces a novel approach to data communication over the Global System for Mobile Communications (GSM) voice channel. It is based on the concept of "symbols"—a set of predefined signals with finite bandwidths. Data are encoded into the symbols, and the symbols are voice coded as they were speech, modulated into the GSM signal, sent over the air, GSM demodulated, voice decoded, and converted back to data. The symbols are synthesized by a genetic algorithm with the aim of maintaining separability after passing them through the voice codec. This method enables data transfer over communication networks that do not have dedicated data channels and could also be used in conjunction with other data services to balance the system load between data and voice channels, allowing optimization of system resources. We present the full algorithmic structure of the system, which performs data communications over the GSM voice channel, and we also give the results of the performance tests.**

***Index Terms*—Data communications, genetic algorithms (GAs), mobile communications, vocoders.**

I. Introduction

WIRELESS communication has already become a part of our daily lives, and it continues to rapidly expand and evolve, covering greater area and offering broader services. There exists a plethora of different communication network architectures and methodologies such as CDMA, CDMA2000, Universal Mobile Telecommunications System, Global System for Mobile Communications (GSM), etc.. Whereas the majority of the development effort is concentrated on new technologies, older preexisting networks have some practical advantages such as low cost, wide coverage, reliability, and acceptance by many local authorities. Among them, the GSM network is probably the most ubiquitous and internationally accepted. Thus, it would be advantageous to add new and improved functionality to the legacy networks.

One of the most sought-after improvements is to add data transmission ability to originally "voice-only" networks such as GSM. The most common data channel added to the GSM network is the General Packet Radio Service (GPRS) [1]. The GPRS data traffic, which is also known as the 2.5 generation wireless, and the GSM voice traffic share the same radio channel [2]. To avoid damage to the quality of service (QoS) for GSM voice traffic, voice is given priority, and only the residual capacity of the network is left for GPRS. Therefore, GPRS available resources are dependent on the voice traffic load, which dominates the GSM network and has an unpredictable (random) nature [2], [3]. This means that queuing delay and outage probability grow with increasing voice traffic, leading to poor QoS for GPRS [3], [5], [8]. Many existing GSM networks have only limited capacity for GPRS, and operators usually need to invest in extending it [4], which is quite costly and may not always be economically feasible. The capacity for data transfer can be significantly increased by moving to the enhanced GPRS, which involves the upgrading of the GSM network with Enhanced Data for Global Evolution (EDGE) technology [9]. However, not only does this require further investment, but QoS issues still remain under conditions of very high traffic load [10]. In addition, GPRS has noncontinuous coverage and interoperability issues [2], [6].

These factors can significantly hinder the use of the GSM data channel. A solution to some of these problems would be to use the compressed speech channel itself to transfer data under some conditions and for some select applications. This technology would obviously be of use in those areas where there is a GSM network but no data service coverage, which is most common in rural areas and developing nations. However, in the likely case that there is already GPRS service available, data transmission through the voice channel would still be useful. Under high-load conditions, when there is a significant probability of outage as the GPRS packet is blocked by voice calls, transferring data through the compressed speech channel would provide better QoS for the data transmission (as the voice traffic has higher priority than the data traffic). This would effectively create a subset of data traffic with higher priority, particularly useful for short data transmissions for which it is important that each data packet be transferred as soon as possible. This could be implemented by carriers who may wish to charge more for higher priority data or by operators who wish higher priority on their data streams independent of the carriers (because the system presented here is a simple voice call as far as the carrier is concerned). In addition, in low-load conditions where there is little voice traffic, a system of data transfer through the compressed voice channel could allow carriers to balance the system load between slots allocated to data and those designated as voice only. Hence, if used along with GPRS, data transmission through the compressed voice channel would allow the balance of the network traffic, maintaining the same QoS for both data and voice.

It should be clear that the modem presented here is not envisaged as a global replacement or direct competition for GPRS or other data transmission systems but as a complementary system to be used for certain applications and situations. In general, communicating data over the voice channel is particularly appropriate for mobile low-bit-rate applications

Manuscript received March 29, 2007; revised July 29, 2007 and October 3, 2007. The review of this paper was coordinated by Prof. G. Saulnier.

C. K. LaDue is with Symstream Technology Ltd., Melbourne, Vic. 3166, Australia (e-mail: chris.ladue@symstream.com).

V. V. Sapozhnykov is with Nanoradio Pty Ltd., Melbourne, Vic. 3166, Australia (e-mail: vitaliy.sapozhnykov@nanoradio.com). AQ1

K. S. Fienberg is with St. Anthony Falls Laboratory, University of Minnesota, Minneapolis, MN 55414 USA (e-mail: fienb004@umn.edu).

Digital Object Identifier 10.1109/TVT.2008.912322

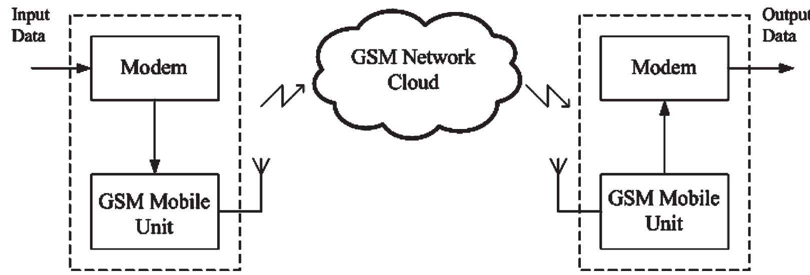

Fig. 1. General system outline.

due to potentially higher QoS (due to higher priority), quicker connection times, and more ubiquitous coverage compared to the dedicated data channels. The last of these, i.e., the wider coverage of GSM voice channel, is particularly important for vehicular applications in which the modem is expected to move to a wide range of locations.

The problem in developing such a system lies in the fact that GSM voice channel is effectively a band-limited nonlinear channel with memory, which is designed for voice-like signals. To allow greater channel capacity, the GSM voice codec extracts the parameters characterizing speech according to the corresponding speech model, and only these parameters are sent over the air [7], [11], [12]. At the receiver side, these parameters are used to generate a replica of the original speech. The compression rate depends on the type of voice codec and varies between 8 and 21.9 [7], [12]. Although to the human ear the regenerated speech sounds very similar to the original, its waveform can in fact be quite different. Hence, it is problematic to achieve reasonable error rates while communicating data over the compressed voice channel because data communication relies on sample-by-sample matching and not on perceptual similarity. An obvious way to compensate for the voice coding impact is to apply equalization [13], [14]. However, this would require the (direct or indirect) identification of the voice codec inverse. The nonlinear nature and differential encoding of the speech parameters make this voice channel practically unidentifiable in terms of structures usually used as equalizers, e.g., finite and infinite impulse response, decision feedback, lattice, etc. [13], [15].

Data communication over the GSM voice channel has a unique set of problems that stipulate a conceptually new approach specifically targeted to the task. The goal of this paper is to design a modem capable of communicating data through such a compressed speech medium. This modem is an addition to the existing GSM system. (Fig. 1).

It converts input data to a pulse-code modulation stream, which is fed into a GSM mobile unit exactly as if it were speech. The GSM mobile unit encodes and modulates this signal as per GSM standard [16] and sends it over the air. The receiver GSM unit demodulates and decodes the received signal, which is, in turn, fed into the modem. The modem outputs estimates of the sent data. Thus, the proposed modem is represented by an encoder/decoder pair that encodes raw data into a signal, which is treated by the GSM system as speech, lets a GSM mobile unit accomplish over-the-air transmission, and decodes this signal back into data. As far as the GSM network is concerned, it is a normal voice call.

There are two design constraints on the signal.

1) It has to pass through the voice channel without raising any alarms (e.g., voice activity detector [7]) and fit into the voice band to avoid distortions caused by filtering.
2) It has to be robust against the GSM voice coder. "Robustness" implies that the signal can be successfully decoded after having been passed through the GSM voice codec.

The general approach to generate a signal, which meets the given constraints, is to apply evolutionary optimization. A genetic algorithm (GA) is used to build the desired signal as a set of waveforms (symbol dictionary). This process is accomplished offline as a modem design procedure. The modem takes these pregenerated symbols and maps the input data onto them. The symbols are concatenated and sent over the air via the GSM unit. On the receiver side, the symbol outputs of the GSM unit are converted back to data. The data estimate is the index of the symbol from the codebook that maximizes the inner product with the received symbol [this method is further referred to as the maximum dot product (MDP) method]. The actual over-the-air transmission is handled by the existing GSM system with its own modulation, forward error correction (FEC), forward error detection (FED), and equalization. Thus, if their correct functioning is assumed, the major source of errors is the voice codec itself.

One approach to utilize the GSM voice channel for secure voice and/or data communications has already been presented in [17]. Katugampala *et al.* described a system where encrypted voice was modulated into a signal using a speech model so that the waveforms possess speech characteristics to minimize distortions caused by speech compression and to avoid voice activity detection (VAD) [18]. These waveforms were then transmitted over the GSM voice channel and demodulated and decrypted at the receiver.

The fundamental difference between the method described in [17] and the method presented in this treatment lies in the way the data are mapped onto the waveform and extracted from it. In [17], these waveforms are speech-like as they are generated using a particular speech model [18]. Thus, at the transmitter, the digital data were "hard coded" into a set of speech-specific parameters according to the speech model and extracted from the waveforms by a speech parameter estimator at the receiver. Therefore, the effects of voice transcoding and VAD were minimized by using a "speech-like" signal. Conversely, in the data transmission method described in this paper, the waveforms used to transmit the data do not necessarily possess speech characteristics or fit any preassumed model.

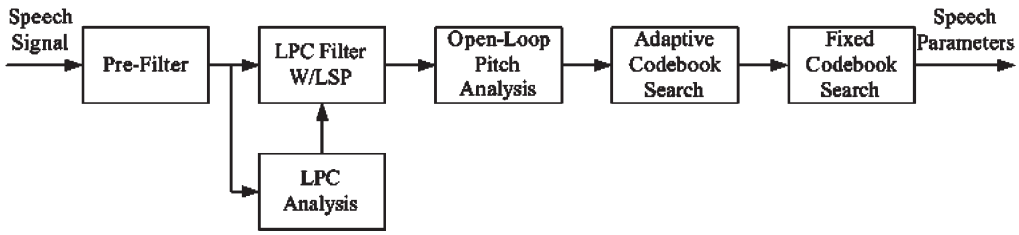

Fig. 2. Generalized ACELP voice-encoding process.

They are generated by the GA to be robust to the voice coding. Distortions caused by the voice-coding/decoding processes are alleviated by maximizing the Euclidian distances between the band-limited waveforms after transmission, and the VAD is avoided by dynamically varying their spectral envelopes. As it will be seen, this leads to a much simpler design, which can easily fit into an embedded system and allows higher data rates.

This paper is organized as follows. Section II presents the theoretical basis of the GSM voice channel. Section III outlines the general modem description, including data-to-symbol encoding and symbol-to-data decoding procedures. Section IV describes the evolutionary symbol-generation process. Sections V and VI present an example implementation and numerical simulations. Section VII discusses the results and considers further implications.

II. GSM Voice Codec

The GSM standard [16] currently supports the following four speech compression techniques: 1) full rate; 2) enhanced full rate (EFR); 3) adaptive multirate; and 4) half rate. The voice codecs, or vocoders, are designed to compress speech signal at the transmitter and accurately regenerate it at the receiver.

The digitized speech signal with a resolution of 13 b and a sampling rate of 8 kHz forms the input to all the GSM speech codecs. The encoder extracts the speech parameters, which are then arranged into a bitstream. The output rate of the speech encoder depends on its type [12]. The parameterized compressed speech signal is encoded and modulated according to the GSM specification and sent over the air. After demodulation, the bits are fed into the speech decoder to synthesize the original speech. In this paper, we consider the GSM EFR voice codec (EFRV). However, the approach should be general enough to be applied to other voice codecs, as will be further discussed in Section VII.

The EFRV is a lossy voice codec that performs mapping between input speech bursts to encoded bit blocks and from encoded bit blocks to reconstructed speech samples, yielding a compression ratio of 8.5 times and a bit rate of 12.2 kb/s for the encoded bitstream. The coding scheme used for compression/decompression is the algebraic code-excited linear prediction (ACELP) coder [19]. The EFRV uses a 10th-order short-term linear prediction and a long-term linear prediction filters. These filters are excited with a combination of adaptive and algebraic codebooks (sets of excitation vectors). The general form of the encoding process is illustrated in Fig. 2 and can be described as follows.

After prefiltering and downscaling, the short-term analysis is performed twice per frame (each 20-ms voice frame comprises four equal subframes). It consists of autocorrelation with two asymmetric windows of 30 ms each, concentrated around every second subframe. The results are converted to short-term lattice filter coefficients and then to line spectral pairs. An open-loop pitch analysis is then performed twice per frame to find two initial estimates of the pitch lag (delay) for each frame. These delays and a grid of delays around them are fed into the speech synthesizer, and its output is compared against the nonsynthesized input. The pitch lags and the gains that correspond to the minimum weighted error are chosen and quantized. The residual signal remaining after the quantization of the adaptive codebook search is modeled by the algebraic codebook using the same approach—synthesize all the possibilities, and choose the one that corresponds to the minimum error. All the calculated speech parameters are packed into the coded speech frame and sent into the GSM physical layer for transmission. The decoding process is represented in Fig. 3.

The parameters of the decoder are set by the values from the received coded speech frame. The speech samples are synthesized by exciting the linear prediction filter with the sum of the algebraic and adaptive codebooks. Although, to a human listener, the reconstructed speech may sound very similar to the original, the waveform is often very different when comparing sample by sample. The vocoder is based on linear prediction and differential encoding of the speech parameters, both of which assume correlation between samples. This assumption holds for voice, which is relatively smooth and slowly changes over time. On the other hand, this is not generally the case for conventional data signals. The more data are transmitted within a fixed bandwidth, the less correlated the samples are. This, in turn, means that the signal fits less into the speech model causing greater distortions and higher error rates. It makes data transmission through the GSM-compressed voice channel a challenge that requires new techniques. As described in Section I, one such technique was presented in [17], but it is quite different compared to our approach, and a comparison of the results is presented in Section VI.

Because of the vocoder's nonlinear nature and memory, it is virtually impossible to represent it by an analytic transfer function. Indeed, the speech-coding process involves multiple quantizations with differential encoding, which causes loss of information about the original signal. The feedback and memory then magnify and propagate these distortions. Moreover, the vocoder output depends not only on the input signal but on its own state as well, which in turn is defined by the previous signal frames including errors. Thus, the concept of conventional

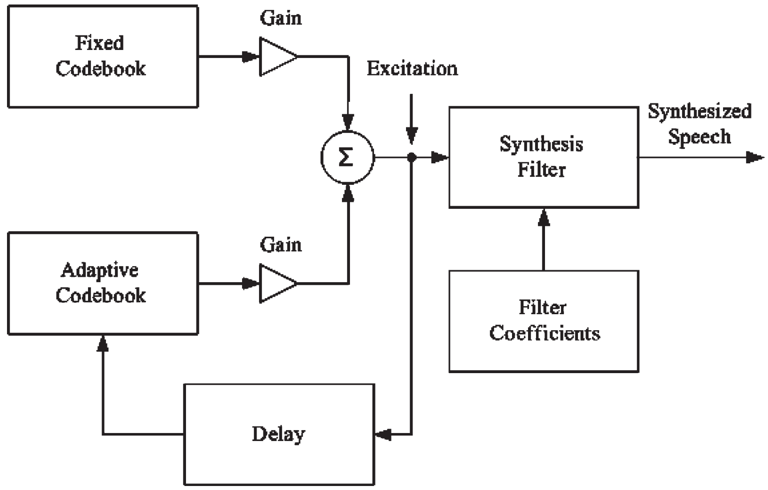

Fig. 3. Generalized ACELP voice-decoding process.

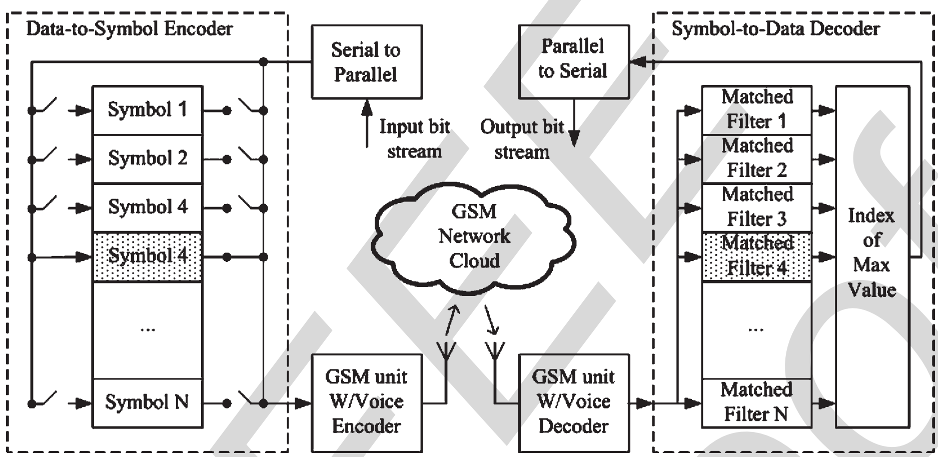

Fig. 4. General modem structure. The "data-to-symbol" encoder is a table lookup that maps the decimal data words onto the symbols. The "symbol-to-data" decoder estimates the sent data words using the MDP method.

equalization (or the "inverse vocoder") is not applicable in this case.

III. General Modem Description

One possible way to enable data transfer through the GSM voice codec is to design a set of waveforms, which are symbols that fit into the voice band ($300-3400$ Hz) and that can be successfully decoded after passing them through the vocoder. The data are mapped onto these symbols on the transmitter side and extracted from them on the receiver side. Thus, this method represents a "data precoding" technique that is followed by the conventional GSM system. In this section, we describe "data-to-symbol" encoding and "symbol-to-data" decoding. A method to generate the symbols will be detailed in Section IV. The general structure of our modem is shown in Fig. 4.

The "data-to-symbol" encoding consists of the following steps.

1) Divide the incoming data bitstream into decimal words i of N_{bit} bits each.
2) Using these words, address the dictionary $\mathbf{D}$, which is the table containing the symbols $\mathbf{s}_i,\ i = 1, 2, \ldots, N_{\mathrm{sym}}$, where $N_{\mathrm{sym}} = 2^{N_{\mathrm{bit}}}$ is the number of symbols in the dictionary. Thus, a single decimal scalar i is encoded into a single vector symbol $\mathbf{s}_i$ by a mapping M given by

$$M : \mathbf{I} \rightarrow \mathbf{D} \tag{1}$$

where

$$\mathbf{I} = \{1, 2, \ldots, N_{\mathrm{sym}}\}$$
$$\mathbf{D} = \{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \ldots, \mathbf{s}_{N_{\mathrm{sym}}}\}. \tag{2}$$

These symbols are scaled to 13-b integers, concatenated, and framed into packets. The packets are then fed into the GSM unit as though they formed an audio signal. In the GSM unit, they are passed through the voice encoder, modulated according to the GSM standard, and sent over the air. On the receiver side, the incoming signal is demodulated and passed through the voice decoder. Then, the GSM unit output is fed into the "symbol-to-data" decoder, which

1) deframes symbol packets and determines the beginning of the first symbol;

2) gets each received symbol and decides which symbol is most likely to have been transmitted. Its index in the dictionary represents the estimate of the transmitted decimal word $\hat{i}$, which in turn is converted to the output data bitstream.

As mentioned above, it is impossible to express the GSM EFRV in terms of transfer function in closed form. On the other hand, the voice-encoding/decoding processes can be represented as an operator given by

$$\mathbf{y} = \mathbf{y}(\mathbf{s}_i, \psi) = V(\mathbf{s}_i, \psi) \tag{3}$$

where $\mathbf{y}$ is the symbol $\mathbf{s}_i$ after the voice-encoding/decoding processes, $\psi = \Psi(\{\mathbf{s}_k\}_{k=1}^{i-1}, \psi_0)$ is the voice codec state, which depends on all previous inputs since the vocoder was reset to its initial state ψ_0, and $V(\cdot)$ represents the vocoder-encoding/decoding processes.

It is convenient to formulate the problem on hand in terms of classical signal detection theory [20]. The transmitter sends digital data encoded into a set of symbols $\{\mathbf{s}_i,\ i = 1, \ldots, N_{\mathrm{sym}}\}$ through the GSM EFRV. The receiver observes the signal $\mathbf{y}$ and tries to decide which $\mathbf{s}_i$ out of N_{sym} symbols was most likely sent. Thus, the symbol-to-data-detection process can be viewed as the *a posteriori* probability distribution maximization

$$\hat{i} = \arg\max_i \left[P(\mathbf{s}_i|\mathbf{y})\right] \tag{4}$$

where $P(\mathbf{s}_i|\mathbf{y})$ is the probability distribution of the sent symbols $\mathbf{s}_i$ given the received symbol $\mathbf{y}$. Because the vocoder output depends not only on the sent symbol but also on its state, the received symbols are not independent. This implies that the *a posteriori* probability distribution should be $P(\mathbf{s}_i|\mathbf{y}, \mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{i-1})$. It is generally possible to model the nonlinear EFRV as a Markov process to estimate the *a posteriori* distribution. However, this would lead to an extremely high complexity of the decision algorithm. To simplify the system design and implementation, we have ignored the previous $\{\mathbf{s}_k\}_{k=1}^{i-2}$ symbols at the cost of some error performance degradation. The decision to truncate the history here is based on the fact that the complexity would exponentially increase with each symbol of the historical series included in the decoding process (combined with the desire to implement the modem in a cheap portable device). In addition, there is a relatively small gain in decoding performance for each additional symbol included. Although there is a dependence on history in the vocoder, the method of the differential coding used means that this dependence has a "long memory" and would have to include the entire history since the last reset to provide significant decoding performance. Hence, we use only $P(\mathbf{s}_i|\mathbf{y})$ for the *a posteriori* probability.

Bayes' rule gives the *a posteriori* probability distribution [20]

$$P(\mathbf{s}_i|\mathbf{y}) = \frac{P(\mathbf{s}_i)P(\mathbf{y}|\mathbf{s}_i)}{P(\mathbf{y})} \tag{5}$$

where $P(\mathbf{s}_i)$ is the *a priori* probability distribution of $\mathbf{s}_i$, $P(\mathbf{y}|\mathbf{s}_i)$ is the likelihood function of $\mathbf{y}$, and $P(\mathbf{y})$ is the *a priori* probability distribution of $\mathbf{y}$. It should be noted that the likelihood function $P(\mathbf{y}|\mathbf{s}_i)$ also does not include previous symbols $\{\mathbf{s}_k\}_{k=1}^{i-2}$ to avoid excessive decoding algorithm complexity. The probability $P(\mathbf{y})$ is independent of the transmitted symbol $\mathbf{s}_i$, so if $P(\mathbf{s}_i)$ is assumed to be uniformly distributed (e.g., can be forced by scrambling), then the symbol detection can be reduced to the maximum likelihood estimation

$$\hat{i} = \arg\max_i \left[P(\mathbf{y}|\mathbf{s}_i)\right]. \tag{6}$$

To proceed, some knowledge of the likelihood function is required. Simulation experiments were used to estimate statistical properties of $P(\mathbf{y}|\mathbf{s}_i)$ by encoding random data into symbols and then passing them through the voice codec and observing the distribution of the received symbols. We discovered that $P(\mathbf{y}|\mathbf{s}_i)$ was ergodic, bell-shaped, and of finite variance. In addition, it was noted that $P(\mathbf{y}|\mathbf{s}_i)$ was not necessarily centered on $\mathbf{s}_i$. However, the MDP estimation does not require the distribution to be inherently centered. It can always be accomplished by subtracting the mean value $\bar{\mathbf{y}}(\mathbf{s}_i)$ given by

$$\bar{\mathbf{y}}(\mathbf{s}_i) = E\left[\mathbf{y}(\mathbf{s}_i, \psi)\right] \tag{7}$$

where $E[\cdot]$ represents expected value over all the voice codec states ψ.

The process of estimating $\bar{\mathbf{y}}(\mathbf{s}_i)$ involves encoding a statistically sufficient amount of random data into the symbols, performing the voice-encoding/decoding processes, and calculating the ensemble average on a symbol-by-symbol basis. The mean symbol $\bar{\mathbf{y}}(\mathbf{s}_i)$ can be calculated offline because the GSM EFRV is standardized and available in software [21]. Hence, it is possible to factorize (3) by

$$\mathbf{y} = \bar{\mathbf{y}}(\mathbf{s}_i) + \mathbf{n}(\mathbf{s}_i, \psi) \tag{8}$$

where $\mathbf{n}(\mathbf{s}_i, \psi)$ is the irregular part of the received signal that depends on both the current vocoder state and the input symbol. This can be thought of as an "effective noise" caused by the voice-encoding/decoding processes. Due to the fact that $\mathbf{n}(\mathbf{s}_i, \psi)$ depends on the sent symbol $\mathbf{s}_i$, the effective signal-to-noise ratio also depends on $\mathbf{s}_i$ and, therefore, cannot be improved by simply increasing signal power. With these properties, after the removal of the mean value, the likelihood function $P(\mathbf{y}|\mathbf{s}_i)$ can be approximated as a centered Gaussian process, with its variance dependent on $\mathbf{s}_i$ as given by

$$P(\mathbf{y}|\mathbf{s}_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left\{-\frac{\|\mathbf{y}(\mathbf{s}_i, \psi) - \bar{\mathbf{y}}(\mathbf{s}_i)\|^2}{2\sigma_i^2}\right\} \tag{9}$$

where $\|\cdot\|$ denotes Euclidean distance, and σ_i^2 is the "effective" noise variance for the symbol $\mathbf{s}_i$.

Substituting (9) into (6) and simplifying the result gives

$$\hat{i} = \arg\min_i \left[\|\mathbf{y}(\mathbf{s}_i, \psi) - \bar{\mathbf{y}}(\mathbf{s}_i)\|^2\right]. \tag{10}$$

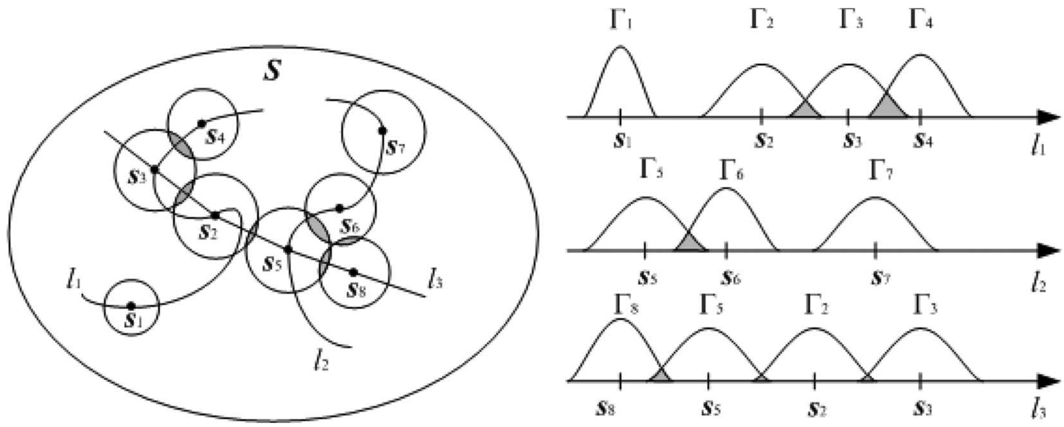

Fig. 5. Symbol space. Points s_i illustrate the transmitted symbols in the symbol space S. The circles around them show the spread after passing through the vocoder. On the right are the probability distributions Γ_i along the lines l_i.

If all $\mathbf{s}_i$ have equal power (this will be achieved by a design constraint), then

$$\hat{i} = \arg\max_{i} \left[\langle \mathbf{y}(\mathbf{s}_i, \psi), \bar{\mathbf{y}}(\mathbf{s}_i) \rangle \right] \tag{11}$$

where $\langle , \rangle$ represents vector dot product.

Thus, the MDP "symbol-to-data"-decoding process involves calculating the dot products of the received symbol $\mathbf{y}(\mathbf{s}_i, \psi)$ and all of the precalculated mean symbols $\bar{\mathbf{y}}(\mathbf{s}_i)$. The index corresponding to the maximum symbol dot product is the estimate $\hat{i}$ of the transmitted data word i. The MDP detector can be implemented by a bank of matched filters or correlators [20].

Having described the data-to-symbol encoding and symbol-to-data decoding techniques, we are ready to design the symbols themselves.

IV. Symbol Design

A. *Symbol Design as a Search Problem*

The symbol-generation process can be formulated as a search problem. The symbols are generated in the frequency domain so that their frequencies are not allowed to be outside of the designated bandwidth and their power is normalized. At the same time, the symbols should be robust enough to still be identifiable after passing through the voice codec. This robustness is a relative measure that can be expressed in terms of error rate. Thus, band limits and unity power form constraints on the search space, whereas robustness represents a cost function.

The search space S for the optimization problem can be defined as all discrete symbols of length N_{sam} samples in time whose frequency spectrum Φ is contained in the frequency interval $[F_{\min}, F_{\max}]$, i.e., $\Phi \in [F_{\min}, F_{\max}]$. Frequencies $F_{\min}$ and $F_{\max}$ should be within the voice band $(300-3400\ \text{Hz})$, but their actual values are dictated by specific carrier requirements. The search space S is also constrained by symbol power P_{sym}. The total power of each symbol is normalized.

The most obvious cost function $C(i)$ for the ith symbol is

$$C(i) = \frac{L_{\text{err}}(i)}{L_{\text{total}}(i)} \tag{12}$$

where $L_{\text{err}}(i)$ is the number of erroneous detections out of $L_{\text{total}}(i)$ times that the ith symbol was sent over the vocoder. In practice, the cost function for all symbols is calculated by encoding a large amount of random data into the symbols, passing them through the voice codec (including both voice-encoding/decoding processes), applying MDP symbol-to-data decoding, and comparing the number of symbol misdetections with the total number of times that each particular symbol passed through the voice codec.

With the search space and cost function defined, we have a constrained minimization problem. Due to the discrete nature of the symbols, the search space is finite dimensional with a dimension proportional to the number of samples in a symbol N_{sam}. Hence, the search space can potentially have a large number of dimensions.

The cost function is defined by the output of the vocoder and is difficult to analytically express. However, there are some known features that the cost function possesses.

1) Nonlinearity comes directly from the vocoder properties.
2) Cost functions $C(i)$ for the different symbols will not be independent because of the MDP decoding (which implies that the more similar the symbols are, the more difficult they are to distinguish). In addition, the vocoder has a memory, thus, the effect on the current symbol depends on the symbols before it.

Fig. 5 illustrates the vocoder impact on the symbols. The plane S represents a multidimensional search space that accommodates symbols $\mathbf{s}_1 = (s_{11}, s_{12}, \ldots, s_{1N\text{sam}})$, $\mathbf{s}_2 = (s_{21}, s_{22}, \ldots, s_{2N\text{sam}}), \ldots, \mathbf{s}_8 = (s_{81}, s_{82}, \ldots, s_{8N\text{sam}})$ as points. The circles around each point encompass 99% of all possible vocoder outputs for each corresponding input symbol. The gray area corresponds to the error probability P_{err} between neighboring symbols. It is more convenient to visualize the symbol mutual impact along curves l_i, each of which goes through the symbols of interest, and $i = 1, \ldots, L$, where L is the number of all possible symbol combinations. Γ_i denote probabilities of symbol occurrence after passing the signal through the voice codec. Errors occur where these probability distributions overlap. Therefore, the greater the overlap, the higher the cost function $C(i)$. It should be noticed that an error (misdetection) can potentially happen between any symbols from the symbol set. Our goal is to find a symbol set from the search space such that probability distributions of the symbols—members of this set—have minimal overlap. This can be accomplished by cost function minimization.

The symbol-generating process is possible to accomplish offline because the EFRV is standardized and available in software.

B. GA

Synthesizing a set of waveforms (symbol dictionary), which satisfies the design specifications mentioned above, is equivalent to finding a group of points in the signal hyperspace where each point represents a symbol. Given that a symbol could be a multidimensional vector, the search hyperspace might be large and multimodal. Thus, an efficient search algorithm is necessary to choose appropriate symbols from the solution space. Possible candidates are hill climbing, gradient descent, exhaustive search, and simulated annealing [13], [22], [23]. The gradient descent algorithm is known to be very efficient in a unimodal search space when the gradient is available. Neither of these two conditions hold in our case. Hill climbing, although it does not require explicit gradient information, is not suitable for finding global extrema in a multimodal space due to its high probability of being trapped in local optima. Thus, if the search space is complicated and multimodal, gradient descent and hill climbing are not appropriate search algorithms for our purpose. Exhaustive search, although not getting stuck in the local optima, is not practical in large search spaces. In addition, simulated annealing is capable of moving out of local minima but heavily depends on the initial conditions. If the initial conditions are not properly chosen, it may not be possible for simulated annealing to converge to the global optimum. Furthermore, simulated annealing requires some smoothness of the search space. In the current problem, the search space is multidimensional and multimodal, and its exact form is unknown and determined by the action of the nonlinear lossy GSM voice codec. Thus, the search algorithms described above are not the best candidates.

A GA is a global optimization technique based on the theory of Darwinian evolution and computer science [24], [25]. A GA is capable of exploring large multimodal search hypersurfaces with unknown structures. It applies a set of specific selection rules to evolve a population of individuals (possible solution vectors) to a state that maximizes the "fitness" (or minimizes the cost function) [24], [25]. It is known to have been successfully applied to complex numerical problems that cannot be analytically solved. These include multimodal function optimization, pattern recognition, and parameter adaptation for complex structures such as neural networks of different architectures [23], [25]–[27]. A GA maintains a population of individuals as a set of searching points and concurrently explores the search spaces along different directions. By applying genetic operators such as crossover and mutation, a GA is capable of moving an individual from one area on the search hypersurface to another, possibly distant, area. This makes it unlikely for the population to get stuck in local extrema. Although a GA does not guarantee the global convergence, if properly parameterized, it can greatly increase the probability of finding the global optimum. Thus, in this paper, the GA is chosen to perform the symbol dictionary generation.

The general GA framework—a set of special "genetic" operators—evolves a population of individuals leading to adaptation by natural selection. Thus, optimization by the GA can be expressed in the following form. First, a population of individuals is selected at random from the entire solution space. Hence, the population comprises a set of possible solutions to the problem at hand. The fitness of each entity is assessed, where the fitness represents the cost function being solved (or its inverse). The population is then sorted based on the fitness to ensure that the fitter individuals are more likely to be selected for mating. The set of genetic operators such as crossover (combining) and mutation (random alteration) is applied to the selected individuals to produce new individuals or offspring, which are added to the population. The fitness is reassessed, and the least fit entities are removed from the pool. This process of assessment, selection, offspring generation, and removal is repeated until the desired performance has been achieved.

C. Evolutionary Design Procedure

If the conventional competitive GA were applied to the problem on hand, then an entire symbol dictionary would be encoded into a single individual (member of the population). These individuals would compete among themselves based on the error rate they provided when loaded into the modem. Then, after the evolution is finished, the "fittest" individual would be the final dictionary for the modem. However, each individual would have $N_{\mathrm{sym}} \times N_{\mathrm{sam}}$ parameters, which potentially could be large. Because of this high dimensionality, the number of $N_{\mathrm{sym}} \times N_{\mathrm{sam}}$ individuals should also be very large to adequately cover the search space. To calculate the fitness of each individual, each corresponding modem would need to be simulated transmitting a statistically significant amount of data through the voice codec. Although symbol generation is an offline process, it could require a significant amount of time.

Hence, to reduce complexity of symbol generation, an alternative form of GA is used. In the cooperative GA [27], [28], the entire population constitutes a single dictionary, with each member of the population comprising a single symbol. The result of the process is not a single optimal individual but is instead a population of individuals coadapted to complement one another, with the whole population representing the symbol dictionary. To successfully develop a symbol dictionary in this way, each individual symbol must evolve to occupy its own "niche"—that is, it must not only perform well in its own right (in this case, transfer through the voice codec with minimal distortion) but must also be different enough from the other symbols so that they can be reliably distinguished by the MDP symbol decoder. How to develop this cooperative behavior between members of a population is known as the "niching problem" in GA literature [29]. In this paper, the evolution of individuals that occupy their own niche is achieved by choosing a GA fitness function that implicitly favors those individuals that are different from the other symbols. This was the approach to the cooperative GA taken by Whitehead and Choate [27]. Keeping this in mind, we are ready to outline the symbol-generation procedure. The process of generating symbols consists of the four steps described as follows.

1) Generate initial symbol set.
2) Select fittest symbols.
3) Produce new symbols, and update symbol set.
4) Iterate epochs: Repeat steps B and C until a sufficiently optimal symbol set is produced, or a maximum number of iterations is exceeded.

1) Step A: Generate an Initial Symbol Set: According to the chosen cooperative GA strategy, the entire population S of individuals $\mathbf{s}_i,\ i = 1, 2, \ldots, N_{\mathrm{sym}}$ constitutes a single symbol set. Each individual $\mathbf{s}_i$ consists of N_{sam} time samples. The symbols are generated in the frequency domain in such a way that they are real in the time domain. For the signal to be real in the time domain, it is required that the following conditions hold [15]:

$$\begin{cases} \mathrm{Re}[\mathbf{\Phi}] & \text{is even} \\ \mathrm{Im}[\mathbf{\Phi}] & \text{is odd} \end{cases} \tag{13}$$

where $\mathbf{\Phi}$ is the complex symbol spectrum. Hence, the symbol-generation procedure can be described as follows.

1) Randomly select a set of N_f complex numbers G_k, $k = 1, 2, \ldots, N_f$, from $\Re^2 \supset [-1, 1] \otimes [-1, 1]$ space. These numbers are used to produce the complex spectrum Φ of the symbol and represent its active frequency load from dc (exclusive) to the Nyquist frequency. If N_f is less than the Fourier bin corresponding to the Nyquist frequency N_N, then the unused $N_N - N_f$ bins are zero padded.
2) Construct $2N_f$ spectral components of Φ using the numbers G_k so that

$$\Phi_i = \begin{cases} 0, & \text{if i} = 0 \\ G_i, & \text{if i} = 1, \ldots, N_f \\ 0, & \text{if i} = N_f + 1, \ldots, N_N \\ G^*_{2N_N - i+1}, & \text{if i} = N_N + 1, \ldots, 2N_N \end{cases} \tag{14}$$

where the asterisk denotes a complex conjugate.
3) Apply the inverse discrete Fourier transform (DFT^{-1}) to Φ to find the time-domain representation

$$\tilde{\mathrm{s}} = \mathrm{DFT}^{-1}(\Phi). \tag{15}$$

4) Normalize the symbol power to produce the final time-domain symbol

$$\mathrm{s} = \frac{\tilde{\mathrm{s}}}{\|\tilde{\mathrm{s}}\|}. \tag{16}$$

These steps are repeated until all N_{sym} symbols are generated. This process is similar to that in the *orthogonal frequency division multiplex* [30]. This way of generating a symbol guarantees that it fits into the designated frequency band by design.

2) Step B: Select the Fittest Symbols: Selection is a process in which fitter individuals are chosen to produce offspring for the next generation. The fitter symbols have a lower value of the cost function $C(i)$.

The modem is loaded with the dictionary of N_{sym} symbols that form the current population S. A pseudorandom datastream is then encoded into the symbols and sent through the vocoder. On the receiver side, the data are extracted by the MDP decoder, and the cost function $C(i)$ for each ith symbol is calculated according to (12). The selection pressure is introduced for parents only, i.e., the N_{off} least fit parents are removed from the population to make room for the next generation of offspring.

“Ranking Selection” [31] is used to select pairs of individuals from the mating pool that will produce offspring for the next generation. It assigns selection probabilities $P_{\mathrm{sel}}(i)$ based on an individual's rank $R(i)$, ignoring absolute fitness value. The individuals in the mating pool are sorted according to their fitness and then assigned a count $R(i)$ that is simply their position in the sorted list. The best individual receives rank 1, the second best receives 2, and so on. The selection probability $P_{\mathrm{sel}}(i)$ is then specified as

$$P_{\mathrm{sel}}(i) = C_s(1 - C_s)^{R(i)-1} \tag{17}$$

where C_s is a constant, such that $0 < C_s < 1$, which controls the slope of the P_{sel} distribution. The closer C_s is to 1, the more P_{sel} is weighted toward the fitter individuals, $i = 1, 2, \ldots, N_{sym}$. Then, parents are selected by one roulette wheel rotation. For each N_{off} offspring, two parents are produced.

3) Step C: Produce New Symbols and Update the Symbol Set: The new symbol-generation process consists of two parts: 1) *crossover* and 2) *mutation*. Crossover and mutation operate on the frequency-domain representation of the symbols to ensure that their spectra do not extend beyond the band limits.

1) *Crossover*: The new offspring is produced by a crossover process in which new individuals are generated by exchanging features of the selected parents. When the parents are represented by vectors of real numbers (real-coded GA), blend crossover BLX $- \alpha$ [32] is known to provide a satisfactory combination of exploration and exploitation. Let us label two chosen parents $\mathbf{G}^1$ and $\mathbf{G}^2$, where $\mathbf{G}^i = \{G^i_k, k = 1, 2, \ldots, N_f\}$, $i = 1, 2$. Then, the offspring $\mathbf{G}'$ is generated by the BLX $- \alpha$ crossover as follows:

$$\mathbf{G}' = \gamma \cdot \mathbf{G}^1 + (1 - \gamma) \cdot \mathbf{G}^2 \tag{18}$$

where γ is a uniformly distributed random variable on the interval $[-\alpha, 1 + \alpha]$. In this case, α is chosen to be 0.5.
2) *Mutation*: The mutation process keeps the diversity of a population and promotes the search in the solution space that cannot be represented by the individuals of the current population. According to this process, a single element G'_k, $k = 1, 2, \ldots, N_f$ of the offspring $\mathbf{G}'$ frequency representation, which is chosen at random, is replaced by a random complex variable from $\Re^2 \supset [-1, 1] \otimes [-1, 1]$.

Both of the genetic operators used in the offspring generation have nonunity probabilities of occurrence. The probability of crossover P_{cross} and the probability of mutation P_{mut} are usually empirically chosen [25], [33]–[35]. When the crossover is not applied, the first parent $\mathbf{G}^1$ is directly copied into the offspring $\mathbf{G}'$. Then, the mutation operator is applied (or not) as usual.

Generated offspring $\mathbf{G}'$ is then used to construct the time-domain symbols $\mathbf{s}_i$ using (14)–(16). After the new offsprings have been generated, they replace the N_{off} least fit parents in the population.

4) Step D: Iterate Epochs: Steps B and C are repeated until one of the following conditions is met: A target error rate C_t is achieved or the maximum number of epochs N_{epoch} have passed. The number of epochs has to be chosen large enough to allow GA convergence. Although replacing a large fraction of the population at each epoch allows rapid evolution, it stipulates a large residual error and can cause instability. To address this issue, a variation of the "elitism" strategy is applied [22]. The number of offsprings generated every epoch N_{off} was reduced by one every N_{reduct} epochs, thereby minimizing the residual error and providing "smoother" adaptation. For proper operation, this requires the number of epochs N_{epoch} to be equal to the initial number of offsprings multiplied by N_{reduct}, so at the final stage of operation (last N_{reduct} generations), there is one offspring being replaced every generation.

If there is no obvious target error rate for the application or if it should simply be as low as possible, then the algorithm is allowed to iterate for the full N_{epoch}. Alternatively, the symbol-generation process can be stopped if there has not been any improvement for a certain time period.

D. Data Rate, Population Size, and Codebook Diversity

Let us denote the system data rate by R, which can be calculated as follows:

$$R = f_s \cdot \eta \tag{19}$$

where f_s is the sampling frequency, and η is the coefficient that determines the codebook properties. This coefficient will be further referred to as the codebook diversity coefficient (CDC), and it is calculated as a ratio between the number of bits encoded into one symbol N_{bit} and the number of samples per symbol N_{sam} given by

$$\eta = \frac{N_{\text{bit}}}{N_{\text{sam}}} = \frac{\log_2(N_{\text{sym}})}{N_{\text{sam}}}. \tag{20}$$

It should be noted that some of the modem parameters (e.g., sampling frequency f_s and designated bandwidth B) are specified by the GSM voice band and are fixed for all designs. Hence, for the system under development, the data rate is completely defined by the CDC.

As it can be seen in (19) and (20), the same CDC (and, therefore, the same data rate) may be achieved by varying the $(N_{\text{bit}}, N_{\text{sam}})$ pair. For example, to build a system with data rate 4000 b/s, the CDC must be equal to 1/2. This ratio may be 6/12 (6 b encoded into one 12-sample symbol), 7/14 (7 b encoded into one 14-sample symbol), 8/16 (8 b encoded into one 16-sample symbol), and so on, leading to 64×12, 128×14 and 256×16 codebooks, respectively. It will be shown in Section V that the codebook (or the population) size affects both the system performance and complexity. Thus, the goal is to identify the codebook dimension (or a set of them) for each data rate, which provides a tradeoff between the modem performance and complexity. It should be noted that because the number of bits per symbol N_{bit} (and, therefore, the population size N_{sym}) is the alphabet parameter responsible for the system data rate and complexity, it is not a completely free GA parameter as it is in the case of a conventional GA [25], [33], [34]. This means that for a particular data rate R, the population size N_{sym} can only take values of 2, 4, 8, 16, and so on until it reaches design constraints on the modem complexity. In addition, N_{sym} has to always be "balanced" by an appropriate symbol length N_{sam} to keep the target data rate constant. The number of samples per symbol is, in turn, determined by the number of active frequencies N_f used to construct the symbol. The symbol active frequency load N_f is the important design parameter as it defines the eigenstructure of the symbol dictionary and, therefore, the dimensionality of the search space S.

Thus, the population (dictionary) size N_{sym} and the symbol length N_{sam} need to be chosen according to practical concerns, with data rate, error performance, and complexity all being taken into consideration.

TABLE I
ALPHABET DESIGN PARAMETERS

$R, b/s$	η	N_{bit}	$N_{sym} \times N_{sam}$	$\Delta f, Hz$	N_f
2000	$\frac{1}{4}$	2	4×8	1000	3
		3	8×12	666.7	4
		4	16×16	500	6
		5	32×20	400	8
		6	64×24	333.3	9
		7	128×28	285.7	10
		8	256×32	250	12
		9	512×36	222.2	13
4000	$\frac{1}{2}$	4	16×8	1000	3
		5	32×10	800	4
		6	64×12	666.7	4
		7	128×14	571.4	5
		8	256×16	500	6
		9	512×18	444.4	7

TABLE II
COOPERATIVE GA PARAMETERS

C_s	P_{cross}	P_{mut}	N_{epoch}
0.075	0.9	0.1	26000

V. SIMULATIONS

To assess the performance of our modem, we have generated a set of dictionaries corresponding to different data rates using different sets of design parameters. The alphabet design parameters are summarized in Table I, and the cooperative GA parameters are represented in Table II.

For all alphabets, the sampling frequency $f_s = 8000$ Hz and the designated bandwidth $B = [300, 3400]$ Hz were used. The frequency spacing Δf and the number of active frequencies N_f were chosen in such a way that the symbol spectrum Φ would not extend beyond the designated frequency band B given by

$$\Phi \le B. \tag{21}$$

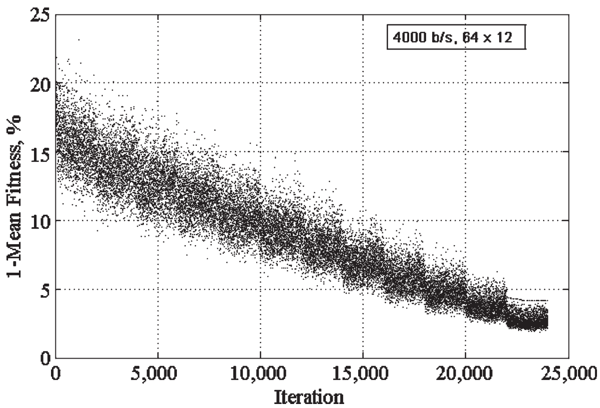

Fig. 6. Mean cost function convergence.

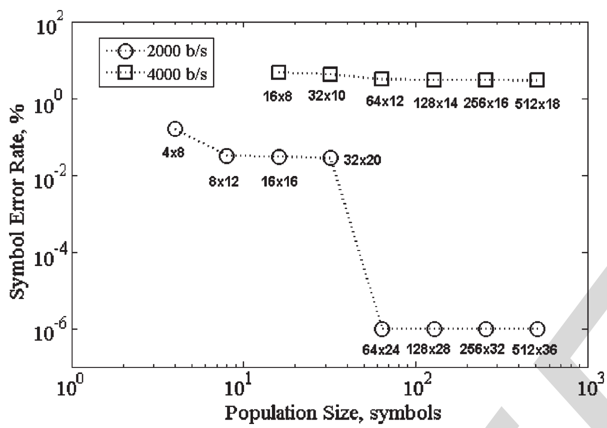

Fig. 7. SER as a function of the population size.

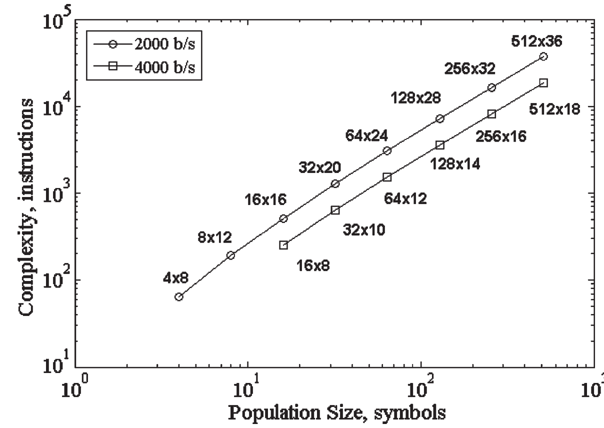

Fig. 8. Receiver complexity as a function of the population size.

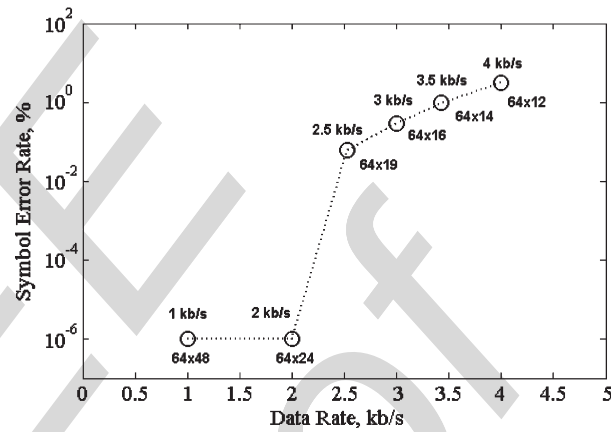

Fig. 9. SER as a function of the data rate.

All the symbol dictionaries were produced following the steps A, B, C, and D described in the Section IV. There were 26 000 epochs to generate each alphabet. The number of offsprings was initially set to 12 and then decremented once in 2000 iterations.

Fig. 6 shows the convergence curve of the mean cost function (which is one minus the mean fitness) generated for the data rate equal to 4000 b/s and a codebook size of 64 symbols by 12 samples. It took approximately from 22 000 to 25 000 iterations for the GA to converge to its error floor. It will be seen that for the fixed population size, systems with lower data rates have lower error floors. In addition, the cost function has a variance decreasing with time and becomes more and more "stair-like" as the elitism strategy reduces the number of generated offsprings every 2000 iterations. It was noted that the best fitness reached its maximum in a few iteration and remained constant (or insignificantly changed) as the elitism strategy preserved the best fit symbols.

Fig. 7 shows the error performance of the proposed modem as a function of the population (codebook) size. The overall modem error performance is determined by the CDC η. With the CDC increased, the data rate R is also increased, but the symbol error rate (SER) gets worse. Conversely, with the CDC decreased, the data rate decreases, but the SER improves. This result is intuitive and may be explained by the symbol diversity concept—the longer the symbol (and, therefore, the richer its frequency content) given the population (codebook) size, the easier it is for the cooperative GA to maximize the Euclidian distances between the symbols and, hence, to minimize the SER. At the same time, the longer the symbol, given the codebook size, the lower the data rate, as the same number of bits is encoded into the longer symbols. Note that for the codebook of the data rate of 4000 b/s, the error performance slightly improves with the increase in the population size. The SER reaches its floor of approximately 3% at a population size of 64 symbols (64×12 alphabet), and a further increase in the population size does not bring any significant SER improvements. On the other hand, for the codebook of the data rate of 2000 b/s, increasing the population size to 64 symbols and above stipulates an SER better than 10^{-6} (as 10^6 symbols were used to evaluate the performance of each alphabet).

The complexity of the receiver as a function of the population (codebook) size is depicted in Fig. 8. As the receiver complexity grows as a power law of the population size, it is clear that the codebook should be kept as compact as possible given the target error rate.

Fig. 9 shows the error performance of the proposed data communication system as a function of the data rate. The system error performance gets worse with the data rate increased. The reason for that, as was mentioned above, is in the reduction of the search space dimensionality—the shorter the symbol, given the fixed alphabet size, the harder it is for the GA to minimize the overlap between the codewords.

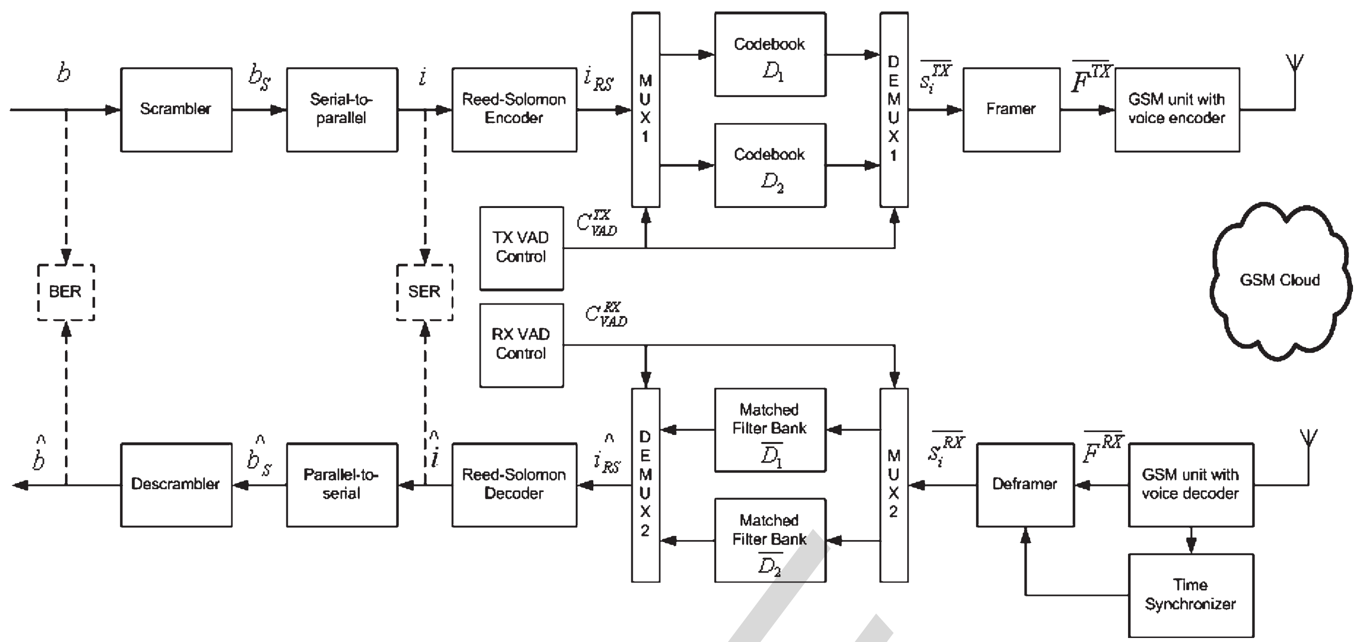

Fig. 10. Example system architecture that enables data transfer through the GSM-compressed voice channel.

In general, the system simulation results suggest the following.

1) The SER grows with the increased data rate.
2) The modem receiver complexity grows as a power law of the codebook size. Hence, there is a tradeoff between the modem receiver complexity and error performance.
3) An increase in the population (codebook) size brings some error performance improvement before reaching a certain limit where no further increase in it impacts the SER.
4) To design a modem that provides data rates greater than 2000 b/s, some forward error correction should be in order. An appropriate FEC scheme for the system under development will be detailed in Section VI.

The simulations have also shown that the system performance was not sensitive to the cooperative GA parameters, so the authors followed the suggestions outlined in [25] and [33]–[35].

As mentioned above, the data modem described in this treatment uses the GSM voice channel as a backbone. The GSM physical layer has its own modulation, equalization, error detection, and FEC [7], [16] to alleviate the impact of multipath, noise, and interference, leaving the voice transcoding as the major source of errors. Nevertheless, it is clear that the overall performance of the modem under consideration is upper limited by the GSM voice channel performance. A very detailed treatment on the GSM/GPRS performance may be found in [6].

VI. System Implementation Example

A. System Architecture

To use the developed modem in a real GSM environment, it is necessary to add some more components to the system. Fig. 10 illustrates an example system that enables data transfer over the GSM-compressed voice channel.

On the transmitter side, the input bitstream b is scrambled in the scrambler block to produce a randomized bitstream b_s. The scrambled bits b_s are fed into the serial-to-parallel block that converts it into $N_{\rm bit}$-bit words i. The Reed–Solomon encoder [36]–[38] buffers $K_{\rm RS}$ $N_{\rm bit}$-bit payload words i and adds $T_{\rm RS}$ parity words to produce a Reed–Solomon codeword $i_{\rm RS}$ of length $M_{\rm RS} = K_{\rm RS} + T_{\rm RS} = 2^{N_{\rm bit}} - 1$ $N_{\rm bit}$-bit words. This enables the correction of $T_{\rm RS}/2$ word errors in each Reed–Solomon codeword $i_{\rm RS}$. Each $N_{\rm bit}$ sample of the Reed–Solomon codeword $i_{\rm RS}$ is used to address one of the codebooks $\mathbf{D_1}$ or $\mathbf{D_2}$, depending on the state of the transmit multiplexer/demultiplexer pair MUX1/DEMUX1. The TX multiplexer/demultiplexer pair is controlled by the transmit TX VAD control block that progressively changes its output $C_{\rm VAD}^{\rm TX}$ from "1" to "0" and *vice versa* every $T_{\rm VAD} = 80$ ms. Therefore, each Reed–Solomon-encoded $N_{\rm bit}$-tuple is encoded into a symbol $s_i^{\rm TX}$. These symbols are shifted into the Framer block where they are combined into packets F^{TX}, and a predefined preamble is added to each frame for synchronization purposes. The frames of the symbols are then passed into the GSM unit for over-the-air transmission. On the receiver side, the output of the GSM unit is the resynthesized symbol frame $F^{\rm RX}$, which is the input to the frame synchronization unit that searches for the predefined preamble. As soon as the beginning of the frame is found, the deframer block is enabled, and the synchronization preamble is removed from the frame to produce data symbols $s_i^{\rm RX}$. The matched-filter bank $\mathbf{\bar{D}_1}$ or $\mathbf{\bar{D}_2}$ is given a received symbol $s_i^{\rm RX}$, depending on the state of the receive multiplexer/demultiplexer pair MUX2/DEMUX2, whose function is similar to the transmit pair. The matched-filter banks have the corresponding transfer functions $\mathbf{\bar{y}_1}$ and $\mathbf{\bar{y}_2}$ and implement the MDP search to produce data estimate $\hat{i}_{\rm RS}$. The mean symbols $\mathbf{\bar{y}_1}$ and $\mathbf{\bar{y}_2}$ were calculated offline during the dictionary-generation process [see (7)]. The estimated data words $\hat{i}_{\mathrm{RS}}$ are then used by the Reed–Solomon decoder, which attempts to correct errors and outputs the estimates $\hat{\hat{i}}$ of the input data words. The parallel-to-serial block converts the output data words $\hat{i}$ into the scrambled bits $\hat{b}_s$. The bitstream $\hat{b}_s$ is then passed to the descrambler block to produce the output bitstream $\hat{b}$. To assess the modem performance, we use the known input data to calculate the symbol and/or BER.

The Reed–Solomon source coding is the intuitive choice for FEC because it operates on N_{bit}-bit words rather than bitstreams. This complies with the fact that each symbol encodes N_{bit} bits of data; thus, errors occur in N_{bit}-bit blocks instead of being evenly distributed. Thus, the ability of Reed–Solomon codes to correct errors in N_{bit}-bit words makes it an efficient FEC method for this application.

B. VAD

In addition to being robust to the voice codec, the designed signal should not raise any alarms in the GSM system. The GSM VAD is a technique designed to avoid transmission when there is no speech. The VAD constantly monitors the signal activity to determine if speech is present or if it is simply just AQ5 noise. If it concludes that there is no speech, it cancels transmission. This can cause problems for data transmission through the GSM channel because it possesses noise-like features.

To ensure that the VAD indicates that there is voice present, it is sufficient to dynamically vary the spectral envelope of the signal over a time scale of approximately $T_{\mathrm{vad}} = 80$ ms, which is the time interval that the VAD engine uses to gather statistics about the speech signal. To implement this, the transmitter can dynamically switch (once every T_{vad}) between two symbol dictionaries designed to have different spectral shapes. Of course, this means that the same switching procedure is synchronously performed on the receiver side. Dictionaries with different spectral shapes can be generated by varying the active Fourier bins [see (14)]. The response of the VAD was from the direct observation of the VAD flag in the software simulation of the AQ6 GSM vocoder system.

C. Prototype System Test

The system described above was implemented and successfully tested over the air. The prototype mobile unit was developed as a custom-designed single-board computer with a host processor based on the ARM920T embedded core and a Sony Ericsson GSM module. The modem used the following parameters: dictionary size $N_{\mathrm{sym}} = 64$ symbols, symbol length $N_{\mathrm{sam}} = 12$ samples, number of active carriers $N_f = 3$, and raw data rate $R = 4000$ b/s, with the rest of the modem parameters the same as in Section V. The Reed–Solomon code parameters were chosen to be $M_{\mathrm{RS}} = 63$, $K_{\mathrm{RS}} = 45$, and $T_{\mathrm{RS}} = 18$. This means that the FEC overhead was 30% and correcting capabilities were 15%. For a 40-min call of continues data transmission (approximately 6720 kb of random data), it produced 26 erroneous bits, giving a BER of 4×10^{-6}. The data transmission did not cause any alarm (such as VAD) to be triggered, so as far as the GSM system was aware, it was a normal voice call.

As already mentioned, once generated, the symbol set is loaded into the modem and needs no more changes. Thus, the dictionary generation can be performed offline, and the complexity of this process is not of a high priority. Therefore, the symbols can be generated using any high-level modeling languages such as Matlab to avoid unnecessary development effort.

D. Performance and Complexity Analysis

Although the system development is performed offline, to be used in a real communication system such as a GSM data modem, the complexity of the receiver must allow real-time embedded system implementation. It should be noted that the data-to-symbol-encoding process is merely a table lookup, whereas the symbol-to-data decoding involves MDP symbol detection. Thus, the complexity of the encoding process is negligible compared to the decoding. Hence, the detection algorithm complexity and code efficiency in terms of millions of instructions per second is imperative. AQ7

Below is the performance and complexity comparisons between the systems described in [17] and in this paper. According to [17], a real-time prototype was implemented to demonstrate one-way encrypted communications over the GSM-compressed voice channel using a single transmitter/receiver pair. The transmitter/receiver pair required two 2-GHz Intel processor-based Windows computers, each having 2 GB of random access memory and a Creative Sound Blaster Audigy soundcard. Both computers were interfaced to GSM Nokia handsets—one for the transmitter and one for the receiver. A raw (uncoded) data rate of 3 kb/s has been reported with a 2.9% BER. Adding error correcting codes (rate 1/2 convolutional codes) yielded a throughput of 1.2 kb/s with a 0.03% BER.

On the other hand, the GSM data modem described in this treatment has been implemented as a portable device based on one 50-MHz ARM920T embedded processor and one Sony Ericsson GSM unit. This handheld GSM modem provided full-duplex communications through the compressed GSM voice channel with an uncoded data rate of 4 kb/s and an SER of less than 3%. The Reed–Solomon coded data rate was 2.8 kb/s with a BER of better than $4 \cdot 10^{-6}$. The software was run on an ARM processor that included an operating system, a proprietary medium access control/link layer, drivers, and the application program. Thus, the modem presented here produced a higher data rate, less errors, and a significantly lower complexity than the system described in [17] for data transfer through the GSM voice channel.

It should be noted that there is a tradeoff between the data rate, error rate, and complexity. For example, to improve the error rate, the symbol length N_{sam} should be increased (see Fig. 9). However, this would reduce the data rate for a given dictionary size N_{sym}. In response, increasing the dictionary size would increase the search time of the MDP decoder, leading to a higher computational burden.

VII. Discussion and Conclusion

In this paper, we have described a new method to transfer data through the compressed GSM voice channel, which involves the evolutionary synthesis of the signal. The chosen application to demonstrate this design concept was to communicate data through the GSM voice channel. Simulations have shown that it was possible to achieve data communications through this highly nonlinear system. The next logical step, following the simulations, was to implement a real modem for over-the-air communications. A portable device implementing the modem presented here was developed and successfully tested in Australia on the three major GSM networks, yielding a raw data rate of 4 kb/s.

A GSM modem of the type described here would be useful in applications such as telemetry, secure communications, wireless automatic teller machines, and faxes—virtually anywhere—where the regular transmission of low-rate data bursts is required. A great advantage of this type of system is that a data service can be deployed where there is no cabling or wireless data channel. Using only a voice channel leads to low setup and maintenance costs and independence from the carrier. For the carrier, the data transmission is just a normal voice call. This may be seen as a "virtual network," or a "network within network," providing new services without the need to upgrade the carrier network. Even when there is a dedicated data channel, e.g., GPRS, already in place, a system to transfer data through the voice channel would allow greater options to balance the system load or create a higher priority datastream for lower data rate but "mission-critical" applications. The GSM voice channel was chosen for its ubiquity, low cost, and great coverage, but the type of modem developed here potentially could be used to communicate data through any compressed voice channel.

Although in this paper we have confined our investigations to the GSM EFRV, it is envisaged that this approach should be applicable to other voice codecs. Such a modem would be designed by simply replacing the EFRV by the other codec in the evolutionary optimization of symbols to produce a set of symbols optimized to be robust through transmission through the new codec. The optimization procedure described here will find such a symbol set if the cost function for the new codec is not significantly more complex than that produced by the EFRV. Whereas this conjecture is unproven, we believe it to hold due to the similarity in the mathematical form of the speech models in commonly used codecs. Initial tests with other voice codecs, such as the GSM half-rate vocoder [7], [12], have been successful, but this work is still ongoing.

The central innovation guiding this work is the evolutionary adaptation to the compressed voice channel, which is impossible to invert due to its nonlinear lossy nature. This represents a new framework for the communication system design. The signal set is adapted to the voice codec by natural selection. The general concept of "adaptation to fit the channel" could potentially be applied to any receiver structure and signal type.

REFERENCES

[1] 3GPP TS 43.064, Overall Description of the GPRS Radio Interface; Stage 2.

[2] M. F. Madkour, "Effect of high GSM voice traffic on GPRS data network and the proposed solutions," in *Proc. 46th IEEE Int. MWSCAS*, Dec. 27–30, 2003, vol. 3, pp. 1259–1262.

[3] S. Ni and S. Haggman, "GPRS perfomance estimation in GSM circuit switched services and GPRS shared resource systems," in *Proc. IEEE WCNC*, Sep. 1999, pp. 1417–1421.

[4] I. Goetz, "Keeping up with GPRS," *Commun. Eng.*, vol. 1, no. 2, p. 46, Apr./May 2003.

[5] T. Alanko, M. Kojo, H. Laamanen, M. Lilijeberg, M. Moilanen, and K. Raatikainen, "Measured performance of data transmission over cellular networks," Dept. Comput. Sci., Univ. Helsinki, Helsinki, Finland, Rep. C-1994-53, 1994.

[6] T. Halonen, J. Romero, and J. Melero, *GSM, GPRS and EDGE Performance*, 2nd ed. Chichester, U.K.: Wiley, 2003.

[7] S. Redl, M. Weber, and M. W. Oliphant, *GSM and Personal Communications Handbook*. Norwood, MA: Artech House, 1998.

[8] H.-H. Liu and J. L. C. Wu, "Delay analysis of integrated voice and data service for GPRS," *IEEE Commun. Lett.*, vol. 6, no. 8, pp. 319–321, Aug. 2002.

[9] S. Hamiti, M. Hakaste, M. Moisio, N. Nefedov, E. Nikula, and H. Vilpponen, "EDGE circuit switched data—An enhancement of GSM data services," in *Proc. IEEE WCNC*, 1999, pp. 1437–1441.

[10] R. Mullner, C. F. Ball, K. Ivanov, F. Treml, and G. Spring, "Quality of service in GPRS/EDGE mobile radio networks," in *Proc. IEEE 59th Veh. Technol. Conf.*, 2004, vol. 5, pp. 2507–2511.

[11] *Digital Cellular Telecommunications Systems (Phase 2+); Enhanced Full Rate (EFR) Speech Transcoding; (GSM 06.60 Version 8.0.1. Release 1999)*, 2000. Available: ETSI EN 300 726.

[12] R. Meston, *Sorting Through GSM Codecs: A Tutorial*. Irvine, CA: Racal Instruments, Jul. 2003. [Online]. Available: http://www.commsdesign.com/showArticle.jhtml?articleID=16501605

[13] D. G. Manolakis, V. K. Ingle, and S. M. Kogon, *Statistical and Adaptive Signal Processing*. New York: McGraw-Hill, 2000.

[14] S. Haykin, *Communication Systems*, 4th ed. New York: Wiley, 2001.

[15] C. L. Phillips and J. M. Parr, *Signals, Systems and Transforms*. Upper Saddle River, NJ: Prentice-Hall, 1995.

[16] *Digital Cellular Telecommunications Systems (Phase 2+); Physical Layer on the Radio Path; General Description; (GSM 05.01 Version 7.1.0. Release 1998)*, 1998. Available: ETSI TS 100 573.

[17] N. N. Katugampala, K. T. Al-Naimi, S. Villette, and A. M. Kondoz, "Real time data transmission over GSM voice channel for secure voice and data applications," in *Proc. 2nd IEE Secure Mobile Commun. Forum: Exploring Tech. Challenges Secure GSM WLAN (Ref. No. 2004/10660)*, Sep. 23, 2004, pp. 7/1–7/4.

[18] N. N. Katugampala, S. Villette, and A. M. Kondoz, "Secure voice over GSM and other low bit rate systems," in *Proc. IEE Secure GSM Beyond: End to End Security Mobile Commun.*, London, U.K., Feb. 2003, pp. 3/1–3/4.

[19] M. R. Schroeder and B. S. Atal, "Code exited linear prediction (CELP): High quality speech at very low bit rates," in *Proc. ICASSP*, 1985, pp. 937–940.

[20] J. Proakis, *Digital Communications*, 4th ed. New York: McGraw-Hill, 2001.

[21] *Digital Cellular Telecommunications Systems (Phase 2+); ANSI-C Code for the GSM Enhanced Full Rate (EFR) Speech Codec; (GSM 06.53 Version 8.0.1. Release 1999)*, 2000. Available: ETSI EN 300 724.

[22] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*. Hoboken, NJ: Wiley, 2004.

[23] B. Bhanu, Y. Lin, and K. Krawiec, *Evolutionary Synthesis of Pattern Recognition Systems*. New York: Springer-Verlag, 2005.

[24] J. Holland, *Adaptation in Natural and Artificial Systems*, 2nd ed. Cambridge, MA: MIT Press, 1992.

[25] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA: Addison-Wesley, 1989.

[26] J. D. Schaffer, L. D. Whitley, and L. J. Eshelman, "Combinations of genetic algorithms and neural networks: A survey of the state of the art," in *Proc. Int. Workshop Combination Genetic Algorithms Neural Netw.*, L. D. Whitley and J. D. Schaffer, Eds., 1992, pp. 1–37.

[27] B. A. Whitehead and T. D. Choate, "Cooperative–competitive genetic evolution of radial basis function centers and widths for time series prediction," *IEEE Trans. Neural Netw.*, vol. 7, no. 4, pp. 869–880, Jul. 1996.

[28] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Proc. 3rd PPSN*, 1994, pp. 249–257.

[29] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp. 42–50.

[30] L. Hanzo, M. Münster, B. J. Choi, and T. Keller, "OFDM and MC-CDMA for broadband multi-user communications," in *WLANs and Broadcasting*. Chichester, U.K.: Wiley, 2003.

[31] F. Hoffmeister and T. Bäck, "Extended selection mechanism in genetic algorithms," in *Proc. 4th Int. Conf. Genetic Algorithms*, 1991, pp. 92–99.

[32] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata," in *Foundation of Genetic Algorithms*, vol. 2, L. D. Whitley, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 187–202.

[33] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-16, no. 1, pp. 122–128, Jan./Feb. 1986.

[34] R. L. Haupt, "Optimum population size and mutation rate for a simple real genetic algorithm that optimizes array factors," in *Proc. IEEE Antennas Propag. Soc. Int. Symp.*, Jul. 16–21, 2000, vol. 2, pp. 1034–1037.

[35] K. A. De Jong, "Analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Univ. Michigan, Ann Arbor, M1, 1975.

[36] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[37] T. K. Moon, *Error Correction Codes: Mathematical Methods and Algorithms*. Hoboken, NJ: Wiley, 2005.

[38] S. B. Wicker, *Error Control Systems for Digital Communications and Storage*. Upper Saddle River, NJ: Prentice-Hall, 1995.

AQ8

Christoph K. LaDue was born on November 15, 1951. He received the degree from San Jose State University, San Jose, CA, in 1982.

He was a Research Scientist for a range of companies in the United States and Australia. He is currently a Principal Researcher with Symstream Technology Ltd., Melbourne, Australia. He is the holder of numerous patents that relate to modern technology including wireless communications, image processing, and artificial intelligence. His broad scientific interests include wireless communications, multidimensional signal processing, multiple-antenna systems, and artificial intelligence.

Vitaliy V. Sapozhnykov was born in Kharkov, Ukraine, on January 4, 1970. He received the B.S.E. (with first-class honors and a gold medal award) and Ph.D. degrees from Kharkov State University, in 1992 and 1996, respectively, both in electrical engineering. AQ9

Until 2006, he held senior research positions for various companies developing advanced communication technologies. He is currently a Senior Algorithm Developer with Nanoradio Ltd., Melbourne, Australia. His research interests span the broad area of digital multidimensional signal processing, reconstructive computerized tomography, synthetic aperture radars, adaptive systems, and stochastic optimization.

Kurt S. Fienberg was born in Gosford, Australia, in 1976. He received the B.Sc. degree (with first-class honors) with majors in physics and mathematics and the Ph.D. degree in remote sensing and climatology from the University of Tasmania, Hobart, Australia, in 1998 and 2003, respectively.

From 2004 to 2006, he was with Symstream Technology, Ltd., Melbourne, Australia, where he worked on algorithm development and signal processing for telecommunication technologies. He is currently a Postdoctoral Research Associate with St. Anthony Falls Laboratory, University of Minnesota, Minneapolis. His research interests cover a range of areas including remote sensing, radiative transfer, signal processing, adaptive systems, and stochastic optimization. He is also interested in the scale invariance and multiscale processes in geophysical fields, including clouds, rainfall, sediment transport, and turbulence, and the effects of the high variability and intermittency of these fields have on remote sensing and modeling.

AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

AQ1 = 3166 was used as the postal [zip codes are US only] code. Please check if correct.
AQ2 = The paragraph starting with "It converts input data..." was connected to the previous paragraph for continuity. Please check if correct.
AQ3 = "problematic" was changed to "difficult." Please check if correct.
AQ4 = "or" was inserted for clarity.
AQ5 = "or simply noise" was changed to "or if it is simply just noise" for clarity. Please check if correct.
AQ6 = "by" was changed to "from the" for clarity. Please check if correct.
AQ7 = "MIPS" was expanded as "million instructions per second." Please check if correct.
AQ8 = Please specify the type of degree earned.
AQ9 = "Kharkov University" was changed to "Kharkov State University." Please check if correct.
Notes: 1) "European Telecommunications Standards Institute" was deleted in Refs. [11], [16] and [21]. Please check.
2) "Los Alamos: IEEE Comput. Society Press" was deleted in Ref. [26]. Please check.
3) "San Francisco, CA: Morgan Kaufmann" was deleted in Ref. [28]. Please check.
4) "San Mateo, CA: Morgan Kaufmann" was deleted in Ref. [31]. Please check.

END OF ALL QUERIES