Distributed turbo coded diversity for relay channel

B. Zhao and M.C. Valenti

A novel coding technique is proposed for the quasi-static fading relay channel. The source broadcasts a recursive code to both relay and destination. The relay decodes, interleaves, and re-encodes the message prior to forwarding. Because the destination receives both codes in parallel, a distributed turbo code is embedded in the relay channel. Results indicate a combined diversity and coding gain, which is significant even for simple constituent codes.

Introduction: A relay channel is a three-terminal network consisting of a source, a relay, and a destination [1]. The source broadcasts a message to both relay and destination, while the relay forwards the message it received to the destination. Relaying has received renewed interest owing to its ability to achieve distributed spatial diversity in wireless networks of single-antenna devices transmitting over quasi-static fading channels [2]. In particular, relaying can be used to form a *virtual antenna array* [3]. Note that it is possible for two devices to cooperate by exchanging their roles as source and relay. This strategy is called *cooperative diversity* [2].

In this Letter, we focus on the problem of coding for the relay channel. A generalisation of our proposed technique may be applied to cooperative diversity. While the relay may use several forwarding strategies, including amplify-and-forward and compress-and-forward [2], we focus on decode-and-forward relaying. In this strategy, the relay decodes the message it received from the source, and then re-encodes it (possibly with a different code) before forwarding. Relay coding has been addressed in [4], which uses distributed rate compatible convolutional codes, and in [5], which uses distributed space time codes. Our approach is to create a distributed *turbo* code by interleaving at the relay [6]. While the approach of [4] and [5] achieve both diversity and coding gain, our approach also benefits from an additional *interleaving* gain (or alternatively, a *turbo processing* gain), and is therefore especially appropriate in the low signal-to-noise ratio regime.

System model: We assume that the relay channel contains very simple devices. In particular, the relay may not simultaneously receive and transmit. Furthermore, neither the source nor the relay knows the relative phase of the other, and therefore they may not transmit coherently. Thus, the relay channel operates in a time division duplex mode. During the first time slot, the source broadcasts to the relay and destination, while in the second slot, just the relay transmits to the destination.

Both the source and the relay generate a very simple code, in this case a two-state rate 1/2 recursive systematic convolutional (RSC) code with octal feedback and feedforward generators (3, 2), respectively. After encoding, the signal is binary phase shift keying (BPSK) modulated. Since the parity output is generated by an accumulator (differential encoder), an alternative interpretation is that each terminal transmits BPSK and differential phase shift keying (DPSK) in parallel (see Fig. 1).

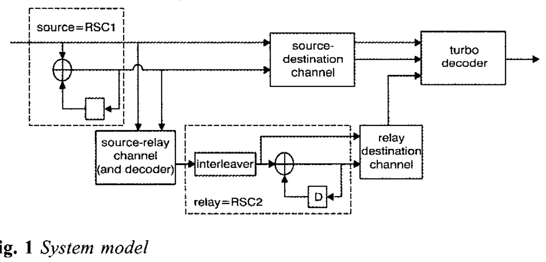
source=RSC1
source-destination channel
turbo decoder
D
source-relay channel (and decoder)
interleaver
relay destination channel
D
relay=RSC2

Fig. 1 *System model*

A conventional relay implementing decode-and-forward will detect the RSC encoded signal and re-encode it with an identical RSC encoder. The destination will receive two versions of the same code word, one directly from the source and the other from the relay (i.e. a repetition code [2]). The two signals may be maximal ratio combined (MRC) and the information bits detected with a Viterbi decoder.

The new twist in our proposed strategy is to add an *interleaver* to the relay, as shown in Fig. 1. If the relay interleaves the estimated source data prior to RSC encoding, then the source and relay have cooperatively constructed a *distributed* turbo code. Recall that with a turbo code, or parallel concatenated convolutional code (PCCC), the data is recursively encoded twice, first in its natural order and again after being interleaved [6]. Thus, the uninterleaved encoding is present in the source-destination path, while the interleaved encoding is present in the relay-destination path. The destination can detect the code iteratively by using a standard turbo decoder [6]. Although the turbo decoder adds some complexity at the destination, the complexity is still reasonable since only two-state encoders are used. While this construction maintains the diversity benefit of relaying, the coding gain is far superior to that of a single RSC observed over two independent channels. This extra coding gain is due to the *interleaving* gain of the turbo code construction and the *turbo processing* gain of the iterative decoder.

Simulation Results: As an illustration, consider a system with a relay located halfway between a source and destination separated by 10 m. Frames of 512 data bits are transmitted at a rate of 1 Mbaud over independent quasi-static Rayleigh fading channels with one-sided noise spectral density $N_o = 10^{-16}$ W/Hz. Assuming a transmit frequency of 2.4 GHz, a path loss coefficient of 3, and a free-space reference distance of 1 m, the average received power at receiver $j \in \{r, d\}$ is $P_j^{(r)} = 10^{-4}(d_{ij})^{-3}P_i^{(t)}$, where d_{ij} is the transmitter-receiver separation and $P_i^{(t)}$ is the transmitted power of node $i \in \{s, r\}$. In a simulation, the transmit power of the source ($P_s^{(t)}$) and relay ($P_r^{(t)}$) were varied independently, and it was noted which ($P_s^{(t)}$, $P_r^{(t)}$) pairs achieved a target source-destination frame error rate (FER) of 10^{-2}. The destination performs up to 15 iterations of decoding, but halts once all errors are corrected. On average, only about two iterations were required.

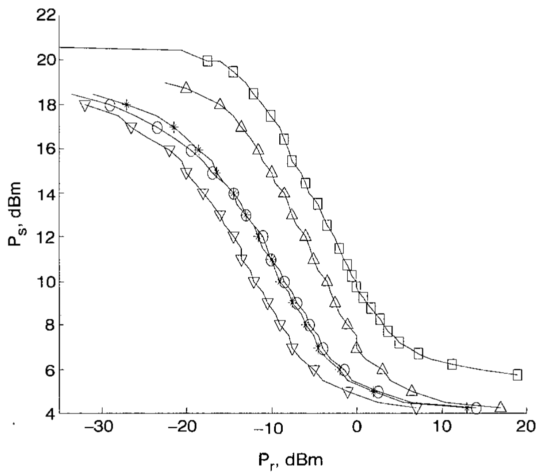

Fig. 2 *Contours showing minimum source and relay transmit powers required to achieve end-to-end FER = 10^{-2} in quasi-static Rayleigh fading relay channel for various codes*
□ uncoded BPSK
△ repetition (RSC) coded
∗ distributed rate 1/4 PCCC
○ distributed rate 1/3 PCCC
▽ distributed rate 1/4 SCCC

Fig. 2 shows a set of contours describing the simulation results for five relaying strategies. The least energy efficient case (upper-right curve) is the uncoded BPSK relay channel. The other four systems are coded, and from least to most power efficient are (1) *Repetition coded*: the source and relay each use a (3, 2) RSC code and the relay does not interleave (the destination performs MRC combining and Viterbi decoding); (2) *Distributed rate 1/4 PCCC*: identical to (1) except the relay interleaves the data prior to RSC encoding and the destination performs iterative (turbo) decoding; (3) *Distributed rate 1/3 PCCC*: identical to (2) except the relay only uses an accumulator (equivalently, the relay's systematic RSC output is punctured); (4) *Distributed rate 1/4 serial concatenated convolutional code (SCCC)*: identical to (3) except the relay re-encodes the data with a (3, 2) RSC encoder prior to interleaving and differential encoding.

The RSC coded system (1) asymptotically has a coding gain of about 1.7 dB over the uncoded BPSK system (since the (3, 2) RSC code has free distance $d_f = 3$). The distributed PCCC provides an additional 4 dB gain over the RSC code. Note that the rate 1/3 and rate 1/4 PCCC codes achieve nearly the same performance, indicating that the relay does not need to transmit its systematic output. The distributed SCCC provides yet another 2 dB gain over the PCCC. While these gains are somewhat modest for a turbo code, the constituent codes only have two states, and hence these codes may be iteratively detected with minimal complexity. However, when the source's transmit power $P_s^{(t)}$ becomes too high or too low, the performance curves of all four coded systems tend to converge and the advantage of distributed turbo coding tends to decrease. On the one hand, when $P_s^{(t)}$ is sufficiently high, no relay power is necessary, and thus the channel only conveys a single RSC code. On the other hand, if $P_s^{(t)}$ is too low, the source-relay link is no longer reliable and the relay will not forward. To improve the energy efficiency at low $P_s^{(t)}$, either the source could use a more powerful code or the relay could move closer to the source.

Discussion: A distributed turbo coding technique has been proposed and applied to the relay channel. The proposed technique achieves a combined diversity and coding gain. Although simple constituent codes are used, the coding gain is significant because of the combination of interleaving at the relay and iterative decoding at the destination. Extension to the cooperative diversity case [2] is straightforward. While only a single relay has been considered, the proposed approach can be extended to the multiple relay case [3]. In this case, a *multiple* turbo code will be embedded in the channel [7]. If feedback were permitted, then the system could be made adaptive by only requiring the relay(s) to transmit when the destination did not receive the message broadcast by the source [2]. This suggests the potential for further improvement by combining the proposed strategy with higher layer mechanisms, such as automatic repeat request (ARQ).

Acknowledgment: This research has been supported by the Office of Naval Research under Grant N00014-00-1-0655.

© IEE 2003 *27 February 2003*
Electronics Letters Online No: 20030526
DOI: 10.1049/el:20030526

B. Zhao and M.C. Valenti (*Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV 26506-6109, USA*)

E-mail: mvalenti@wvu.edu

References

1 COVER, T.M., and EL GAMAL, A.A.: 'Capacity theorems for the relay channel', *IEEE Trans. Inform. Theory*, 1979, **25**, (5), pp. 572–584
2 LANEMAN, J.N.: 'Cooperative diversity in wireless networks: algorithms and architectures'. PhD Dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, August 2002
3 GASTPAR, M., KRAMER, G., and GUPTA, P.: 'The multiple-relay channel: coding and antenna-clustering capacity'. ISIT, Laussane, Switzerland, 2002, p. 136
4 HUNTER, T., and NOSRATINIA, A.: 'Cooperative diversity through coding'. ISIT, Laussane, Switzerland, 2002, p. 220
5 LANEMAN, J.N., and WORNELL, G.W.: 'Distributed space-time coded protocols for exploiting cooperative diversity in wireless networks'. GLOBECOM, Taipei, Taiwan, 2002
6 BERROU, C., GLAVIEUX, A., and THITIMASJSHIMA, P.: 'Near Shannon limit error-correcting coding and decoding: turbo-codes'. ICC, Geneva, Switzerland, 1993, pp. 1064–1070
7 DIVSALAR, D., and POLLARA, F.: 'Multiple turbo codes'. MILCOM, San Diego, CA, USA, 1995, pp. 279–285

Selection of weight quantisation accuracy for radial basis function neural network using stochastic sensitivity measure

Wing W.Y. Ng and D.S. Yeung

Minimising the number of bits per connection weight in hardware realisation of a radial basis function neural network (RBFNN) will result in high-speed and low-cost implementation, with possible increase in output error. A weight quantisation accuracy selection method is proposed, to find an appropriate number of bits for a given stochastic sensitivity measure, which quantifies the relationship between the variance of the output error and first- and second-order statistics of input, weight and their perturbations.

Introduction: There is a trade-off in the hardware implementation of a neural network. Too many bits may lead to a slow and expensive system, whereas too few bits may cause higher error. In [1] Piché used the stochastic sensitivity measure to analyse the error involved in the hardware realisation of the Madalines. Yeung and Sun [2] and Zeng and Yeung [3, 4] extended the investigation to the multilayer perceptrons. In [2] an analytical formula for a stochastic sensitivity of a radial basis function neural network (RBFNN) was derived. In this Letter we relax a number of restrictive assumptions stated in [1–4], thus making the hardware realisation of an RBFNN more practical. The stochastic sensitivity of an RBFNN is defined as the variance of the output error with respect to statistics of weight, input and their perturbations.

Stochastic model and sensitivity measure of RBFNN: In considering the hardware implementation of an RBFNN, the network is assumed fully connected and the maximum number of hidden neurons is specified. The weights will not be specified until it is applied to a particular pattern recognition task. As pointed out in [1], during the design phase of the circuit, all possible different RBFNNs rather than a specific one should be considered, and it is more appropriate to analyse an ensemble of RBFNNs.

We assume that the ensemble of weights and weight perturbations are identically independently distributed (i.i.d.) random variables with uniform distribution. The uniform distribution is to ensure that every possible value is considered with an equal probability. Although for some applications the weights may be more concentrated in certain regions but overall it will be i.i.d. We also make the assumption as in [1] to include external (first layer) input errors so that all layers in the RBFNN of the ensemble would have input errors. Both the inputs and their perturbations are assumed to be i.i.d. random variables with uniform distribution.

We extend Piché's work on Madalines [1] to an RBFNN with the following restrictions removed:

(i) Large network: this restriction is not suitable for small to medium scaled pattern classification problems.
(ii) Small weight error with zero-mean: this would not allow using less number of bits per weight when high speed performance outweighs low output error.
(iii) Same variance and mean for each weight, weight perturbation, input and input perturbation: by removing this restriction, the designer has the flexibility to choose a different number of bits per weight for different neurons.

We define the output of the kth output neuron as

$$y_k(X + \Delta X) = \sum_{j=1}^{M} (w_{kj} + \Delta w_{kj}) \phi_j(X + \Delta X) + w_{k0} + \Delta w_{k0}$$

where $X = (x_1, x_2, \ldots, x_N)^T \in R^N$ is the network input vector; w_{kj} is the connection weight between the hidden node Φ_j and the kth output neuron; and M is the number of hidden neurons. We also have

$$\phi_j(X) = \exp\left(-\frac{\|X - U_j\|^2}{2v_j^2}\right)$$

where $U_j = (u_{j1}, u_{j2}, \ldots, u_{jN})$ is the centre vector of the jth basis function, and v_j is the width of RBF.