Evolutionary Algorithm Aided Interleaver Design for Serially Concatenated Codes

R. G. Maunder and L. Hanzo

***Abstract*— Previous interleavers designed for Serially Concatenated Codes (SCCs) have achieved a limited Minimum Hamming Distance (MHD) between the legitimate permutations of the encoded bit sequence. Hence, we propose a novel Evolutionary Algorithm (EA) capable of designing improved interleavers for SCCs, without artificially limiting the achievable MHD. As a result, increased MHDs and therefore reduced error floors are achieved, even if only a modest EA complexity can be afforded.**

***Index Terms*— Joint source and channel coding, trellis codes, information rates.**

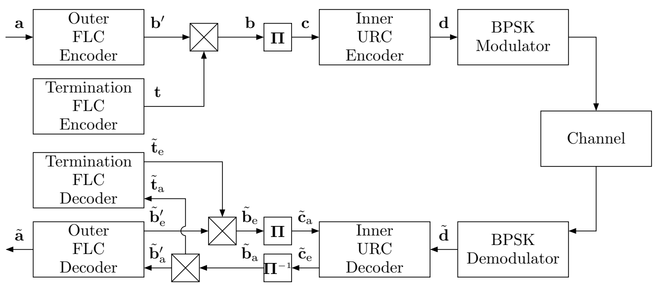
a
Outer FLC Encoder
$\mathbf{b}'$
b
Π
c
Inner URC Encoder
d
BPSK Modulator
Termination FLC Encoder
t
Channel
$\tilde{\mathbf{t}}_{\mathrm{e}}$
Termination FLC Decoder
$\tilde{\mathbf{t}}_{\mathrm{a}}$
$\tilde{\mathbf{b}}'_{\mathrm{e}}$
$\tilde{\mathbf{b}}_{\mathrm{e}}$
$\tilde{\mathbf{c}}_{\mathrm{a}}$
$\tilde{\mathbf{a}}$
Outer FLC Decoder
$\tilde{\mathbf{b}}'_{\mathrm{a}}$
$\tilde{\mathbf{b}}_{\mathrm{a}}$
Π^{-1}
$\tilde{\mathbf{c}}_{\mathrm{e}}$
Inner URC Decoder
$\tilde{\mathbf{d}}$
BPSK Demodulator

Fig. 1. SCC schematic. LLR sequences are indicated by applying a diacritical tilde to the notation of the corresponding bit sequence. A subscript of ‘a’ indicates *a priori* LLRs, while a subscript of ‘e’ is employed for extrinsic LLRs. The boxed crosses indicate either a multiplexing or a demultiplexing operation.

I. Introduction

In Serially Concatenated Codes (SCCs) [1] like that of Fig. 1, the Symbol Error Ratio (SER) is governed by the union bound [2, Equation (2.5)]

$$\mathrm{SER} \leq \sum_{E_{\mathbf{d}}=1}^{N_{\mathbf{d}}} \sum_{E_{\mathbf{a}}=1}^{N_{\mathbf{a}}} \frac{E_{\mathbf{a}} A_{E_{\mathbf{a}},E_{\mathbf{d}}}}{N_{\mathbf{a}}(1+E_0/N_0)^{E_{\mathbf{d}}}}, \tag{1}$$

when the $N_{\mathbf{a}}$-symbol source sequence $\mathbf{a}$ is represented by the $N_{\mathbf{d}}$-bit encoded sequence $\mathbf{d}$ and transmitted over an uncorrelated narrowband Rayleigh fading channel, having a Signal to Noise Ratio (SNR) of E_0/N_0. Here, $A_{E_{\mathbf{a}},E_{\mathbf{d}}}$ [3, Equation (8)] is the component of the distance spectrum corresponding to permutations of $\mathbf{a}$ that are separated by a Hamming distance of $E_{\mathbf{a}}$ and that are mapped to permutations of $\mathbf{d}$ that are separated by a Hamming distance of $E_{\mathbf{d}}$. The bound of (1) is tight at high SNRs [2], where it is dominated by the *Minimum Hamming Distance* (MHD) w, which is the lowest value of $E_{\mathbf{d}}$ that is associated with non-zero distance spectrum components $A_{E_{\mathbf{a}},E_{\mathbf{d}}}$. As a result, we have $\mathrm{SER} \approx m/(1+E_0/N_0)^w$, where $m = \sum_{E_{\mathbf{a}}=1}^{N_{\mathbf{a}}} \frac{E_{\mathbf{a}}}{N_{\mathbf{a}}} A_{E_{\mathbf{a}},w}$ is the *multiplicity* of the MHD. Since the distance spectrum is dictated by the interleaver $\mathbf{\Pi}$ of Fig. 1 [4], this may be beneficially designed to yield a high MHD w having a low multiplicity m and hence to yield a reduced SER at high SNRs.

While the references of [4] have proposed many interleaver designs for increasing the MHD w of Parallel Concatenated Codes (PCCs), these are typically unsuitable for SCCs. This is because, in contrast to PCCs, SCCs typically employ a concatenation of different codes, having different Hamming distance properties. Furthermore, while PCCs are typically linear, in SCCs often non-linear outer codes are employed for joint source and channel coding [5]. In these applications, the error sensitivity and entropy of the source may vary throughout the source sequence $\mathbf{a}$, motivating the employment of an irregular outer encoder [6]. Finally, the input $\mathbf{b}$ of an SCC's interleaver has only a limited set of legitimate permutations, since it is provided by the encoded bit sequence generated by the outer encoder.

Paper approved by A. K. Khandani, the Editor for Coding and Information Theory of the IEEE Communications Society. Manuscript received June 8, 2009; revised June 19, 2010 and December 20, 2010.

The authors are with the School of Electronics and Computer Science, University of Southampton, 4004 Mountbatten Building, Salisbury Road, SO17 1BJ United Kingdom (e-mail: rm@ecs.soton.ac.uk; lh@ecs.soton.ac.uk).

Digital Object Identifier 10.1109/TCOMM.2011.01.xxxxx

Against this backdrop, interleaver designs that can increase the MHD w of SCCs remain scarce. Similarly to S-random interleavers [7], the interleavers proposed in [8], [9] were designed for mitigating the correlation between neighbouring Logarithmic Likelihood Ratios (LLRs) in the extrinsic sequences $\tilde{\mathbf{b}}_{\mathrm{e}}$ and $\tilde{\mathbf{c}}_{\mathrm{e}}$ during iterative decoding. Since this approach increased the associated MHDs to a certain degree [4], the corresponding error floors were slightly improved. By contrast, the Code Matched Interleaver (CMI) of [10] was specifically designed to improve the MHD of a particular SCC, although it was unable to guarantee a MHD above $w = 3$.

In Section II of this letter, we propose a novel Evolutionary Algorithm (EA) for designing the interleaver of an SCC. In contrast to the approaches of [7]–[10], our EA is capable of achieving an MHD w that is limited only by the degree of freedom offered by the interleaver's length $N_{\mathbf{b}}$. By removing all artificial limits on the MHD w, our EA achieves lower error floors than all previous approaches, even if only a modest EA complexity can be afforded, as demonstrated in Section III. Furthermore, our EA may be applied to the entire range of SCCs that is represented by Fig. 1, where Binary Phase Shift Keying (BPSK) is employed. Here, we refer to our outer code as a *Fixed Length Code* (FLC), which we define as an arbitrary linear or non-linear, regular or irregular block code. The inner code is referred to as a *Unity Rate Code* (URC), which we define as an arbitrary regular or irregular convolutional code, provided that it has a coding rate of unity, which is a necessary condition for avoiding capacity loss [11]. Albeit we consider the termination [12] of the URC in this letter, our EA may be readily adapted to the case where the termination code of Fig. 1 is removed. We will therefore conclude in Section IV that our approach may be used in numerous applications, ranging from the conventional SCCs of [1] to the specialised joint source and channel codes of [5].

II. Evolutionary Algorithm

In this section, we detail our EA conceived for designing the interleaver $\mathbf{\Pi}$ of the scheme shown in Fig. 1. In Section II-A, we demonstrate that the components of the distance spectrum $A_{E_\mathbf{a},E_\mathbf{d}}$ can be expressed in terms of the *Woven Error Patterns* (WEPs) [13] that are facilitated by the design of the interleaver $\mathbf{\Pi}$. Section II-B shows that the set of all WEPs can be uniquely and efficiently represented using a tree structure. In Section II-C, we show that an A* search algorithm [14] can be employed to direct the construction of the WEP tree, in order to efficiently identify the WEPs that contribute to the MHD w. Section II-C also shows that a second A* search algorithm can be iteratively employed in order to efficiently determine the pair of indices in the interleaver $\mathbf{\Pi}$ that can be swapped for the sake of yielding the highest attainable improvements in the MHD w, as well as in its multiplicity m and hence ultimately in the attainable SER.

A. Woven Error Patterns

Our EA employs the data structure of Fig. 2 to represent the WEPs that are facilitated by a particular interleaver design $\mathbf{\Pi}$.

Here, each WEP e describes the transformations imposed on the bit sequences $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$, when the FLC input sequence $\mathbf{a}$ is switched from one permutation to another. Naturally, this permutation-switching corresponds to the transformation of some of the FLC codewords within $\mathbf{b}'$, as well as the potential transformation of the termination FLC codeword $\mathbf{t}$ [12]. In order to describe these transformations, a specific FLC *Error Pattern* (EP) $e.f$ in the list $e.\mathbf{f}$ identifies the indices $e.f.\mathbf{i}$ of the toggled bits within each of the transformed codewords in the FLC output sequence $\mathbf{b}$. Furthermore, the toggling of bits in $\mathbf{b}$ causes the toggling of the corresponding interleaved bits in the URC input sequence $\mathbf{c}$. In turn, the toggled bits in the sequence $\mathbf{c}$ are associated with disjoint sequences of toggled bits in the URC output sequence $\mathbf{d}$. A URC EP $e.u$ in the list $e.\mathbf{u}$ therefore identifies the indices $e.u.\mathbf{i}$ of the toggled bits in the sequence $\mathbf{c}$ that are associated with each of these disjoint sequences.

It can be shown that the components of the distance spectrum are given by

$$A_{E_\mathbf{a},E_\mathbf{d}} = \sum_{e\in\mathbf{e}(\mathbf{\Pi})\Big|\substack{e.n=E_\mathbf{a}\\ e.w=E_\mathbf{d}}} e.m, \tag{2}$$

where the notation is defined in Fig. 2 and $\mathbf{e}(\mathbf{\Pi})$ is a set comprising all WEPs that are facilitated by the interleaver $\mathbf{\Pi}$, as well as all possible Combinations of WEPs (CWEPs) that do not interfere with each other [13]. As implied by (2), the total URC output weight $e.w$ of a WEP identifies the Hamming distance between the corresponding pair of legitimate permutations of the URC output sequence $\mathbf{d}$ [4]. Therefore, our EA endeavours to redesign the interleaver $\mathbf{\Pi}$ in order to eliminate the WEPs having the lowest total URC output weight $e.w$, as described in the following sections.

Fig. 2. Data structure employed to construct WEPs in the proposed EA.

Each WEP e comprises the following:

- A list of FLC EPs $e.\mathbf{f}$, each of which $e.f \in e.\mathbf{f}$ affects a different FLC codeword in the FLC output sequence $\mathbf{b}$. Each FLC EP $e.f \in e.\mathbf{f}$ comprises the following:
  - A list of FLC output bit indices $e.f.\mathbf{i}$, which identify the bits of the FLC codeword in $\mathbf{b}$ that are toggled by the FLC EP $e.f$ when the WEP e occurs;
  - A multiplicity $e.f.m$, which quantifies the probability that the FLC codeword will adopt a permutation that is susceptible to the FLC EP $e.f$. Note that when the FLC is non-linear, the FLC EP may not transform all FLC codeword permutations into another legitimate permutation. Also note that different FLC codeword permutations may occur with different probabilities in joint source and channel coding applications.
- a list of URC EPs $e.\mathbf{u}$, each of which $e.u \in e.\mathbf{u}$ affects a different disjoint subset of the URC input sequence $\mathbf{c}$. Each URC EP $e.u \in e.\mathbf{u}$ comprises:
  - A list of URC input bit indices $e.u.\mathbf{i}$, which identify the bits in $\mathbf{c}$ that are toggled by the URC EP $e.u$, when the WEP e occurs;
  - a URC output weight $e.u.w$, which quantifies the number of bits in the URC output sequence $\mathbf{d}$ that are toggled by the URC EP $e.u$ when the WEP e occurs.
- a list of URC input bit index repositories $e.\hat{\mathbf{u}}$, each of which $e.\hat{u} \in e.\hat{\mathbf{u}}$ comprises:
  - a list of so-called *unallocated* URC input bit indices $e.\hat{u}.\mathbf{i}$. While these indices identify bits in $\mathbf{c}$ that are toggled, when the WEP e occurs, URC EPs in $e.\mathbf{u}$ have not yet been created to accommodate them;
  - a minimum value $e.\hat{u}.w^+$ for the total URC output weight that can be obtained by creating (one or more) URC EPs in $e.\mathbf{u}$ to accommodate all of the unallocated URC input bit indices in $e.\hat{u}.\mathbf{i}$. This may be determined using the method of constrained subcodes [15].
- the number $e.n$ of FLC codewords in the sequence $\mathbf{b}'$ that are corrupted by the WEP, which will equal the cardinality $|e.\mathbf{f}|$ if the termination FLC codeword $\mathbf{t}$ is unaffected by the WEP, or $|e.\mathbf{f}|-1$ if it is;
- a multiplicity $e.m = \prod_{e.f\in e.\mathbf{f}} e.f.m$;
- a total URC output weight $e.w = \sum_{e.u\in e.\mathbf{u}} e.u.w$;
- a minimum value $e.w^+ = \sum_{e.u\in e.\mathbf{u}} e.u.w + \sum_{e.\hat{u}\in e.\hat{\mathbf{u}}} e.\hat{u}.w^+$ for the total URC output weight that can be obtained by creating new URC EPs in $e.\mathbf{u}$ to accommodate all of the unallocated URC input bit indices in $e.\hat{\mathbf{u}}$;
- a rank $e.r$, as will be detailed in Section II-B.1.

B. WEP Tree

The proposed EA efficiently identifies and eliminates the WEPs having the lowest total URC output weight with the aid of a tree structure. Here, each leaf node uniquely represents a different one of the WEPs that are facilitated by the interleaver $\mathbf{\Pi}$. In contrast to these so-called *complete* WEPs, each branch node represents a WEP e that is referred to as *incomplete*, since its list of repositories $e.\hat{\mathbf{u}}$ contains some unallocated URC input bit indices, as described in Fig. 2. The unallocated indices of $e.\hat{\mathbf{u}}$ are the basis of the relationship between the corresponding branch node and its children nodes. More specifically, there is a child node for each of the different

WEPs that can be obtained by augmenting the FLC EPs in $e.\mathbf{f}$, in order to eliminate the unallocated URC input bit indices in $e.\hat{\mathbf{u}}$. A child node becomes a leaf, if the augmented FLC EPs eliminate all of the unallocated URC input bit indices without creating any new ones. Otherwise, the child node becomes a branch and the corresponding WEP is said to be incomplete. The WEP tree may be initialised and constructed as described in Sections II-B.1 and II-B.2, respectively.

1) WEP Tree Initialisation: The WEP tree is initialised by creating the nodes at a depth of one. This is achieved using the initialisation function $\mathbf{e}(i_{\text{in}}, \mathbf{\Pi}, \mathbf{r})$, where $\mathbf{\Pi}$ is the interleaver's sequence of bit mappings and the other parameters are detailed below. This function provides a list $\mathbf{e}(i_{\text{in}}, \mathbf{\Pi}, \mathbf{r})$ of incomplete WEPs that reside at a depth of one in the tree. Each of these incomplete WEPs $e \in \mathbf{e}(i_{\text{in}}, \mathbf{\Pi}, \mathbf{r})$ has a list of FLC EPs $e.\mathbf{f}$ containing only a single entry $e.f$, plus a list of unallocated URC input bit index repositories $e.\hat{\mathbf{u}}$ containing only a single entry $e.\hat{u}$ and finally an empty list of URC EPs $e.\mathbf{u}$.

Each call to the initialisation function will only generate WEPs that toggle a particular bit in the URC input sequence $\mathbf{c}$, as specified by the index i_{in} in the function's parameters. The deinterleaver $\mathbf{\Pi}^{-1} = \{\Pi^{-1}[i]\}_{i=1}^{N_{\mathbf{b}}}$ maps the URC input bit having the index i_{in} to the bit in the FLC output sequence $\mathbf{b}$ that has the index $\Pi^{-1}[i_{\text{in}}]$. This FLC output bit will also be toggled by the WEP considered, but this is additionally required to form part of a legitimate FLC EP. The initialisation function therefore considers the set of all valid FLC EPs that contain the FLC output bit having the index $\Pi^{-1}[i_{\text{in}}]$. A different FLC EP from this set provides the single FLC EP $e.f$ for each of the WEPs in the list $e \in \mathbf{e}(i_{\text{in}}, \mathbf{\Pi}, \mathbf{r})$.

Furthermore, the interleaver $\mathbf{\Pi} = \{\Pi[i]\}_{i=1}^{N_{\mathbf{b}}}$ maps each of the FLC output bit indices $e.f.i \in e.f.\mathbf{i}$ in the FLC EP $e.f$ to a particular URC input bit index $\Pi[e.f.i]$. The set of these indices are assigned to the WEP's single unallocated URC input bit index repository $e.\hat{u}$. Note that one of the unallocated bit indices in $e.\hat{u}.\mathbf{i}$ will be equal to the URC input bit index i_{in} that was specified as a parameter of the function call.

As we shall show in Section II-C, the initialisation function $\mathbf{e}(i_{\text{in}}, \mathbf{\Pi}, \mathbf{r})$ is called for every URC input bit index $i_{\text{in}} \in \{1, 2, 3 \ldots N_{\mathbf{b}}\}$ at the commencement of our EA. In order to prevent consecutive calls to the initialisation function from creating redundant copies of the same WEPs, a common ranking list $\mathbf{r} = \{r[i]\}_{i=1}^{N_{\mathbf{b}}}$ is provided as a parameter for each call. This provides a unique rank $r[i]$ for each of the URC input bit indices $i \in \{1, 2, 3 \ldots N_{\mathbf{b}}\}$. The rank $e.r$ of each WEP $e \in \mathbf{e}(i_{\text{in}}, \mathbf{\Pi}, \mathbf{r})$ obtained by the initialisation function is set equal to that of the URC input bit index i_{in} provided in the function's parameters, $e.r = r[i_{\text{in}}]$. During the operation of the initialisation function, a WEP e is only admitted to the list $\mathbf{e}(i_{\text{in}}, \mathbf{\Pi}, \mathbf{r})$ if each of its unallocated URC input bit indices $e.\hat{u}.i \in e.\hat{u}.\mathbf{i}$ has a rank $r[e.\hat{u}.i]$ that satisfies $r[e.\hat{u}.i] \leq e.r$.

2) WEP Tree Construction: A second function is employed to obtain the WEPs, which are represented by branch and leaf nodes at depths of more than one in the tree structure. This function provides a list of child WEPs $\mathbf{e}(e_{\text{in}}, \mathbf{\Pi}, \mathbf{r}, w_{\text{in}})$ by developing the particular parent WEP e_{in} that is provided as a parameter for the function-call. In general, this WEP e_{in} may contain multiple FLC EPs in $e_{\text{in}}.\mathbf{f}$, multiple URC EPs in $e_{\text{in}}.\mathbf{u}$ and multiple unallocated URC input bit index repositories in $e_{\text{in}}.\hat{\mathbf{u}}$. The input WEP's list of FLC EPs $e_{\text{in}}.\mathbf{f}$ forms a subset of the corresponding list $e.\mathbf{f}$ in each WEP $e \in \mathbf{e}(e_{\text{in}}, \mathbf{\Pi}, \mathbf{r}, w_{\text{in}})$ developed. Similarly, the list of URC EPs $e_{\text{in}}.\mathbf{u}$ forms a subset of $e.\mathbf{u}$. However, none of the unallocated URC input bit indices in $e_{\text{in}}.\hat{\mathbf{u}}$ appear in the repositories $e.\hat{\mathbf{u}}$ of any WEP e developed.

Instead, new URC EPs are created in $e.\mathbf{u}$ that (a) accommodate all of the unallocated URC input bit indices in $e_{\text{in}}.\hat{\mathbf{u}}$ and (b) give a total URC output weight $e.w$, which is equal to the value specified by the parameter w_{in}, where $w_{\text{in}} \geq e_{\text{in}}.w^{+}$. Note that each of the newly created URC EPs in $e.\mathbf{u}$ is required to contain at least one of the unallocated URC input bit indices in $e_{\text{in}}.\hat{\mathbf{u}}$. Besides this however, the newly created URC EPs are permitted to invoke additional URC input bit indices that are not included in $e_{\text{in}}.\hat{\mathbf{u}}$. These additional URC input bit indices imply that the WEP e developed toggles additional URC input bits and, following deinterleaving by $\mathbf{\Pi}^{-1}$, additional FLC output bits, as described in Section II-B.1. The development function accommodates the indices of these additional FLC output bits by creating new FLC EPs in $e.\mathbf{f}$, which are similarly permitted to invoke additional FLC output bit indices. Furthermore, the interleaver $\mathbf{\Pi}$ maps these additional FLC output bit indices to corresponding URC input bit indices, which remain unallocated in the WEP e developed and are assigned to $e.\hat{\mathbf{u}}$ accordingly. The development function employs a search proceedure in order to include every possible valid combination of new FLC EPs and URC EPs within the resultant list of WEPs $\mathbf{e}(e_{\text{in}}, \mathbf{\Pi}, \mathbf{r}, w_{\text{in}})$.

Every WEP $e \in \mathbf{e}(e_{\text{in}}, \mathbf{\Pi}, \mathbf{r}, w_{\text{in}})$ provided by the development function is assigned the same rank as the WEP provided by the parameter e_{in}, $e.r = e_{\text{in}}.r$. The development function will reject a WEP e developed if $e.\mathbf{u}$ or $e.\hat{\mathbf{u}}$ include any URC input bit indices that have a rank higher than this. As described in Section II-B.1, this approach prevents the duplication of WEPs. Furthermore, a WEP e developed will be rejected, if any of the bit indices added to $e.\mathbf{u}$, $e.\mathbf{f}$ or $e.\hat{\mathbf{u}}$ either (a) interferes with any of the EPs that were inherited from $e_{\text{in}}.\mathbf{u}$ and $e_{\text{in}}.\mathbf{f}$ or (b) prevents the creation of legitimate EPs in $e.\mathbf{u}$ or $e.\mathbf{f}$ in this and in future calls to the development function. Since there are numerous reasons to reject the WEPs developed, the size of the list $\mathbf{e}(e_{\text{in}}, \mathbf{\Pi}, \mathbf{r}, w_{\text{in}})$ provided by the development function is limited and the complexity of our EA is kept manageable, as we shall demonstrate in Section III.

Note that a WEP e developed will be complete if it is obtained without inserting any unallocated URC input bit indices into $e.\hat{\mathbf{u}}$. Otherwise, the WEP e must be developed further in subsequent iterations of the EA.

C. $A^{\star}$ Algorithms

Our EA employs the data structure and functions of Sections II-A – II-B.2 to design an interleaver $\mathbf{\Pi}$ that satisfies two constraints. Firstly, like an S-random interleaver [7], the interleaver is required to map the bits within each FLC codeword to URC input bits that are separated by at least s other bit positions. This mitigates the correlation between neighbouring LLRs in the extrinsic sequences $\tilde{\mathbf{b}}_{\text{e}}$ and $\tilde{\mathbf{c}}_{\text{e}}$ during iterative decoding, hence improving the convergence of this process [7]. Secondly, the interleaver is required to satisfy the constraints of [12], which allows the termination FLC encoder of Fig. 1 to terminate the trellis of the URC. Within these constraints, our EA attempts to maximise the minimum of the WEPs' total URC output weights w, since this is equal to the MHD between any two permutations of the encoded bit sequence $\mathbf{d}$. The EA's secondary objective is to minimise the total multiplicity m of the WEPs e that have the minimal total URC output weight $e.w = w$. In this way, low error floors can be obtained, as described in Section I. Our EA proceeds according to the pseudo code of Algorithm 1.

Observe that in line 1 of Algorithm 1, the EA is initialised using a random interleaver design $\mathbf{\Pi}$ that satisfies the above-mentioned two constraints. Following this, the EA is entirely deterministic, employing no further randomisations. In lines 2 to 9 of Algorithm 1, the EA uses an $A^\star$ algorithm to efficiently determine the MHD w and multiplicity m that results for the initial interleaver. More specifically, the EA searches for complete WEPs that have a particular total URC output target weight, which is incremented in a loop, until the search becomes successful. During this search, a list of WEPs $\mathbf{e}$ is developed gradually, allowing the results of each iteration in the loop to assist those that follow.

As the EA progresses, the design of the interleaver $\mathbf{\Pi}$ is gradually evolved, with the creation of a new generation occurring in each iteration of the loop between lines 10 and 43. In each generation, WEPs having the minimum total URC output weight w are eliminated in such a way that we only create new WEPs that have a total URC output weight higher than w. This is achieved by carefully selecting two of the interleaver's bit mappings and swapping them, exchanging the two FLC output bits that are mapped to a particular pair of URC input bits.

A second $A^\star$ algorithm is employed to efficiently find the particular bit mapping swap that satisfies the above-mentioned constraints and achieves the highest improvement of the scheme's MHD w and/or multiplicity m. More specifically, the potential improvement that is offered by every possible swap is tentatively assessed in line 12. Starting with the swaps having the greatest potential, the actual improvement (if any) that each offers is determined in the loop between lines 13 and 41. This loop terminates, when none of the remaining swaps has the potential to improve upon the best swap found so far. As a result, the duration of the search employed by each generation of the EA is minimised.

The improvement offered by a particular swap is assessed in lines 18 to 34 of Algorithm 1. This is achieved by searching for complete WEPs that have a particular target total URC output weight, which is incremented in a loop, until the search becomes successful. However, in contrast to when the EA was initialised in lines 2 to 9, it is not necessary to start this search from scratch. Instead, the list of WEPs obtained in the previous generation of the EA may be updated by removing only those WEPs that are directly affected by the swap and replacing them with new WEPs. As a result, the complexity of our proposed EA is minimised.

The proposed EA is terminated in line 43, when no more swaps that improve the scheme's MHD w and/or multiplicity m remain. However, the EA can also be manually terminated before this occurs in order to limit its total run time. In this

Algorithm 1 EA for designing the interleavers of serially concatenated FLC and URC codes.

```
1:  Generate a random interleaver $\mathbf{\Pi}$ that has a length $N_\mathbf{b}$, a separation $s$ [7] and facilitates the termination of the URC [12].
2:  Initialise the ranking list $\mathbf{r} = \{r[i]\}_{i=1}^{N_\mathbf{b}}$ with an arbitrary unique rank for each URC input bit index $i$.
3:  for each $i \in \{1, 2, 3 \ldots N_\mathbf{b}\}$, use the initialisation function to add WEPs to a list $\mathbf{e} \leftarrow \mathbf{e}(i, \mathbf{\Pi}, \mathbf{r})$.
4:  Initialise the MHD $w = 0$.
5:  repeat {the development of the WEP list}
6:      $w = w + 1$.
7:      for each $e \in \mathbf{e}$, if $w \geq e.w^+$ then use the development function to add WEPs to the list $\mathbf{e} \leftarrow \mathbf{e}(e, \mathbf{\Pi}, \mathbf{r}, w)$.
8:  until the subset $\mathrm{comp}(\mathbf{e})$ of $\mathbf{e}$ containing all complete WEPs is not empty.
9:  Set the multiplicity $m = \sum_{e \in \mathrm{comp}(\mathbf{e})} \frac{e.n}{N_\mathbf{a}} e.m$.
10: repeat {the evolution of a new generation}
11:     Initialise the best Hamming distance, multiplicity, interleaver and ranking list found so far $w_\mathrm{best} = w$, $m_\mathrm{best} = m$, $\mathbf{\Pi}_\mathrm{best} = \mathbf{\Pi}$, $\mathbf{r}_\mathrm{best} = \mathbf{r}$.
12:     Assess the minimum multiplicities $\mathbf{M} = \{m[i_1][i_2]\}$ that could result from swapping the two FLC output bits that the interleaver $\mathbf{\Pi}$ maps to each particular pair of URC input bits, having the indices $i_1 > i_2 \in \{1, 2, 3 \ldots N_\mathbf{b}\}$. Here, $m[i_1][i_2]$ may be obtained by summing the multiplicities of every WEP in $\mathrm{comp}(\mathbf{e})$ having a list of URC EPs that does not contain exactly one of the URC input bit indices $i_1$ and $i_2$.
13:     while $\min(\mathbf{M}) = 0$ or ($w = w_\mathrm{best}$ and $\min(\mathbf{M}) < m_\mathrm{best}$) do
14:         Initialise the URC input bit indices $i_1, i_2 = \mathrm{argmin}(\mathbf{M})$.
15:         Initialise a new interleaver, ranking list and WEP list $\mathbf{\Pi}_\mathrm{new} = \mathbf{\Pi}$, $\mathbf{r}_\mathrm{new} = \mathbf{r}$, $\mathbf{e}_\mathrm{new} = \mathbf{e}$.
16:         Swap $\Pi^{-1}_\mathrm{new}[i_1]$ with $\Pi^{-1}_\mathrm{new}[i_2]$ or, equivalently, swap $\Pi_\mathrm{new}[\Pi^{-1}_\mathrm{new}[i_1]]$ with $\Pi_\mathrm{new}[\Pi^{-1}_\mathrm{new}[i_2]]$.
17:         if the swap maintains a separation exceeding $s$ between each pair of URC input bits that the deinterleaver $\mathbf{\Pi}^{-1}$ maps to the same FLC codeword [7] and the swap still facilitates the termination of the URC [12] then {develop the interleaver further}
18:             Set $r_\mathrm{new}[i_1]$ and $r_\mathrm{new}[i_2]$ to unique values greater than $\max(\mathbf{r}_\mathrm{new})$.
19:             Remove each $e \in \mathbf{e}_\mathrm{new}$ if $e.\mathbf{u}$ or $e.\hat{\mathbf{u}}$ contains at least one of $i_1$ or $i_2$.
20:             for both $i \in \{i_1, i_2\}$, use the initialisation function to add WEPs to another list $\mathbf{e}' \leftarrow \mathbf{e}(i, \mathbf{\Pi}_\mathrm{new}, \mathbf{r}_\mathrm{new})$.
21:             Initialise the new Hamming distance $w_\mathrm{new} = 0$.
22:             repeat {the development of the WEP list}
23:                 $w_\mathrm{new} = w_\mathrm{new} + 1$.
24:                 for each $e \in \mathbf{e}'$, if $w_\mathrm{new} \geq e.w^+$ then use the development function to add WEPs to the list $\mathbf{e}' \leftarrow \mathbf{e}(e, \mathbf{\Pi}_\mathrm{new}, \mathbf{r}_\mathrm{new}, w_\mathrm{new})$.
25:             until $w_\mathrm{new} = w$ or the subset $\mathrm{comp}(\mathbf{e}')$ of $\mathbf{e}'$ containing all complete WEPs is not empty.
26:             if $w_\mathrm{new} = w$ then {there is potential for improvement}
27:                 Add the WEPs in $\mathbf{e}'$ to the list $\mathbf{e}_\mathrm{new}$.
28:                 if $\mathrm{comp}(\mathbf{e}_\mathrm{new})$ is empty then
29:                     repeat {the development of the WEP list}
30:                         $w_\mathrm{new} = w_\mathrm{new} + 1$.
31:                         for each $e \in \mathbf{e}_\mathrm{new}$, if $w_\mathrm{new} \geq e.w^+$ then use the development function to add WEPs to the list $\mathbf{e}_\mathrm{new} \leftarrow \mathbf{e}(e, \mathbf{\Pi}_\mathrm{new}, \mathbf{r}_\mathrm{new}, w_\mathrm{new})$.
32:                     until the subset $\mathrm{comp}(\mathbf{e}_\mathrm{new})$ of $\mathbf{e}_\mathrm{new}$ containing all complete WEPs is not empty.
33:                 end if
34:                 Set the multiplicity $m_\mathrm{new} = \sum_{e \in \mathrm{comp}(\mathbf{e}_\mathrm{new})} \frac{e.n}{N_\mathbf{a}} e.m$.
35:                 if $w_\mathrm{new} > w_\mathrm{best}$ or ($w_\mathrm{new} = w_\mathrm{best}$ and $m_\mathrm{new} < m_\mathrm{best}$) then {an improvement has been found}
36:                     $w_\mathrm{best} = w_\mathrm{new}$, $m_\mathrm{best} = m_\mathrm{new}$, $\mathbf{\Pi}_\mathrm{best} = \mathbf{\Pi}_\mathrm{new}$, $\mathbf{r}_\mathrm{best} = \mathbf{r}_\mathrm{new}$.
37:                 end if
38:             end if
39:         end if
40:         $m[i_1][i_2] = m_\mathrm{best}$.
41:     end while
42:     $w = w_\mathrm{best}$, $m = m_\mathrm{best}$, $\mathbf{\Pi} = \mathbf{\Pi}_\mathrm{best}$, $\mathbf{r} = \mathbf{r}_\mathrm{best}$.
43: until no more improvements are found.
```

case, the best interleaver $\mathbf{\Pi}_{\text{best}}$ found so far is output.

III. Simulation Results

In this section, we compare the performance of the scheme shown in Fig. 1, when employing random and S-random designs for the interleaver $\mathbf{\Pi}$, as well as designs obtained using the proposed EA of Section II. We consider the joint source and channel coding of source symbol sequences $\mathbf{a}$ having various lengths in the set $N_{\mathbf{a}} \in \{25, 50, 100, 200\}$, which are typical in challenging Wireless Sensor Network (WSN) scenarios or in speech and audio applications. The FLC encoder of Fig. 1 maps the $K_{\mathbf{a}} = 8$-ary uncorrelated source symbols to $L_{\mathbf{b}'} = 5$-bit FLC codewords, which occur with the unequal probabilities of occurrence $P(\mathtt{11001}) = 0.0402$, $P(\mathtt{01010}) = 0.1066$, $P(\mathtt{11110}) = 0.1615$, $P(\mathtt{10011}) = 0.1917$, $P(\mathtt{01101}) = 0.1917$, $P(\mathtt{00000}) = 0.1615$, $P(\mathtt{10100}) = 0.1066$ and $P(\mathtt{00111}) = 0.0402$. Furthermore, the memory-1 URC and the corresponding $L_{\mathbf{t}} = 3$-bit termination FLC encoder of [12] are employed to obtain the bit sequence $\mathbf{d}$.

Transmission over an uncorrelated narrowband Rayleigh fading channel having various SNR values per bit, E_b/N_0, was simulated. In order to assess their *average* performance, different random and S-random interleaver designs were employed for the transmission of each source symbol sequence. In the case of the S-random designs [7], separations of $s \approx \sqrt{N_{\mathbf{b}}/2} \in \{8, 11, 16, 22\}$ were employed for source sequence lengths of $N_{\mathbf{a}} \in \{25, 50, 100, 200\}$, respectively. By contrast, the same EA-designed interleaver was employed for all source symbol sequences having the same length $N_{\mathbf{a}}$. Here, the same separation of $s = 8$ was employed for each source sequence length $N_{\mathbf{a}}$, since this was found to be sufficient for mitigating the correlation between the extrinsic LLRs, as described in Section II-C. These EA-designed interleavers were obtained by running ten instances of the proposed EA in parallel for four hours on an Beowulf cluster [16]. In this duration, the average number of swaps considered between lines 18 and 38 of Algorithm 1 was found to be approximately six times higher than the value of $N_{\mathbf{a}}$. Of the ten resultant interleaver designs obtained for each value of $N_{\mathbf{a}}$ considered, we selected the one having the most desirable MHD w and multiplicity m. Our SER versus E_b/N_0 results are presented in Fig. 3.

As shown in Fig. 3, the EA-designed interleavers offer less pronounced error floors than the S-random interleavers, which in turn offer lower error floors than the random interleavers. These findings may be explained by the MHDs w, which were found to be 1, 2 and 5 in the case of the random, S-random and EA-designed interleavers, respectively.

In order to quantify how remarkable the MHD of $w = 5$ achieved by the EA is, we conducted an additional simulation. This used lines 2 to 9 of Algorithm 1 to determine the MHD offered by 16 million random interleaver designs, for the case where $N_{\mathbf{a}} = 25$. Of these, none had a MHD higher than that of our design and only 32 exhibited the same MHD of $w = 5$, which is just 0.0002%. However, none of those 32 interleavers exhibited a multiplicity m as low as that of our design, a separation s as high as ours, while facilitating termination, like our design. Indeed, the percentage of all interleaver designs that can be considered to out-perform ours is likely to be significantly lower than 0.0002%.

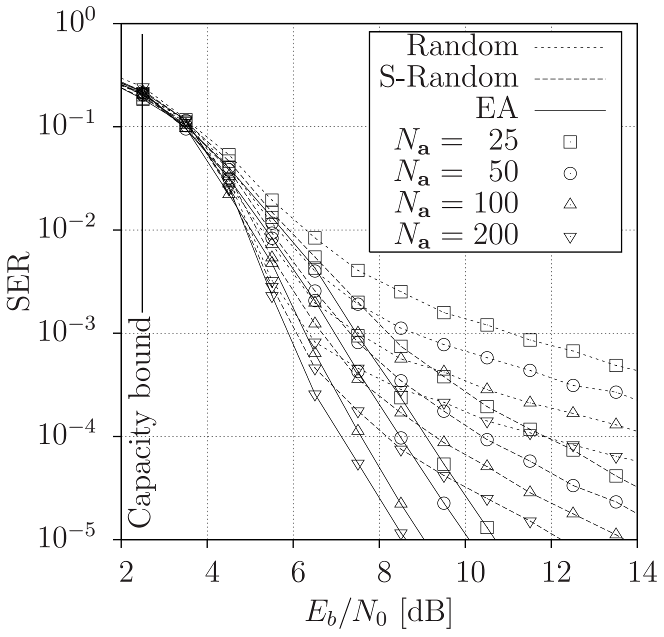

Fig. 3. Comparison of the SER versus E_b/N_0 performance of the scheme shown in Figure 1 when employing random, S-random and EA-designed interleavers for various source symbol sequence lengths $N_{\mathbf{a}}$.

IV. Conclusions

In this letter, we have proposed an EA that can design interleavers for a wide range of SCCs, without imposing an artificial limit on the achievable MHDs w. As a result, our EA can achieve significantly higher MHDs and lower error floors than previous approaches, even if only a modest EA complexity can be afforded. Indeed, Fig. 3 shows that, while random and S-random interleavers can impose error floors at SERs above 10^{-5}, these can be significantly lowered by employing interleavers designed using the proposed EA. As a result, at an SER of 10^{-4}, the EA-designed interleavers offer gains of 1 – 3 dB over S-random interleavers and 5 – 11 dB over random interleavers, depending on the length of the source sequences.

References

[1] S. Benedetto and G. Montorsi, "Iterative decoding of serially concatenated convolutional codes," *Electron. Lett.*, vol. 32, no. 13, pp. 1186–1188, June 1996.

[2] I. Sason and S. Shamai, "Performance analysis of linear codes under maximum-likelihood decoding: A tutorial, foundation and trends," *Foundations Trends Commun. Inf. Theory*, vol. 3, no. 1/2, pp. 1–225, 2006.

[3] X. Jaspar and L. Vandendorpe, "Design and performance analysis of joint source-channel turbo schemes with variable length codes," in *Proc. IEEE Int. Conf. Commun.*, May 2005, pp. 526–530.

[4] B. Vucetic, Y. Li, L. C. Perez, and F. Jiang, "Recent advances in turbo code design and theory," *Proc. IEEE*, vol. 95, no. 6, pp. 1323–1344, June 2007.

[5] J. Hagenauer and N. Görtz, "The turbo principle in joint source-channel coding," in *Proc. IEEE Inform. Theory Workshop*, Mar. 2003, pp. 275–278.

[6] M. Tüchler and J. Hagenauer, "EXIT charts of irregular codes," in *Proc. Conf. Inform. Sci. Syst.*, Mar. 2002, pp. 748–753.

[7] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations," *Telecommun. Data Acquisition Progress Report*, vol. 122, pp. 56–65, Apr. 1995.

[8] F. Kienle and N. Wehn, "Macro interleaver design for bit interleaved coded modulation with low-density parity-check codes," in *Proc. IEEE Veh. Technol. Conf.*, May 2008, pp. 763–766.

[9] J. Yu, M.-L. Boucheret, R. Vallet, A. Duverdier, and G. Mesnager, "Interleaver design for serial concatenated convolutional codes," *IEEE Commun. Lett.*, vol. 8, no. 8, pp. 523–525, Aug. 2004.

[10] N. Ehtiati, A. Ghrayeb, and M. R. Soleymani, "Improved interleaver design for turbo coded intersymbol interference channels," in *Proc. IEEE Veh. Technol. Conf.*, Oct. 2003, vol. 1, pp. 327–331.

[11] J. Kliewer, A. Huebner, and D. J. Costello, "On the achievable extrinsic information of inner decoders in serial concatenation," in *Proc. IEEE Int. Symp. Inf. Theory*, July 2006, pp. 2680–2684.

[12] R. G. Maunder and L. Hanzo, "Iterative decoding convergence and termination of serially concatenated codes," *IEEE Trans. Veh. Technol.*, vol. 59, no. 1, pp. 216–224, Jan. 2010. [Online]. Available: `http://eprints.ecs.soton.ac.uk/16133/`

[13] M. Breiling, "Analysis and design of turbo code interleavers," Ph.D. dissertation, Erlanger Berichte aus Informations und Kommunikationstechnik, 2002.

[14] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybernetics*, vol. 4, no. 2, pp. 100–107, July 1968.

[15] R. Garello, P. Pierleoni, and S. Benedetto, "Computing the free distance of turbo codes and serially concatenated codes with interleavers: Algorithms and applications," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 5, pp. 800–812, May 2001.

[16] T. Sterling, D. J. Becker, D. Savarese, J. E. Dorband, U. A. Ranawake, and C. V. Packer, "Beowulf: A parallel workstation for scientific computation," in *Proc. Int. Conf. Parallel Processing*, Aug. 1995, pp. 11–14.