A Factor-Graph Approach to Universal Decoding

Pascal O. Vontobel

***Abstract*— We consider data communication over a discrete memoryless channel where neither the sender nor the receiver knows the channel law. The setup is universal in the sense that no training sequence is allowed, i.e. no position of the channel code is allowed to be fixed to a certain symbol. We discuss a variety of approaches for solving this problem efficiently. It turns out that it is worthwhile to design decoders which try to minimize the symbol error probability. This is in contrast to the usual approach where the block error probability is minimized.**

As a side result, we present a very simple derivation of the so-called one-sweep algorithm by Johansson and Zigangirov. In the context of universal channel decoding, this algorithm has certain structural advantages over the BCJR algorithm.

I. Introduction

In the last decade it has become more and more clear how one can efficiently achieve reliable communication close to capacity when the channel law is known. A very helpful tool in deriving such codes / decoders has been the factor-graph / message-passing iterative decoding framework.

Some work has also been done for formulating decoders when the channel law is not known, see e.g. [1], [2], [3], [4], [5], [6]. However, in these papers the channel law was never totally unknown (the channel was within a very specific class of channels) and/or the decoders could rely on the presence of training sequences or pilot symbols. In this paper we study the case where the channel law is unknown except that it is a discrete memoryless channel (DMC) with known input and output alphabet.

We remark that papers and books that discuss universal decoding include [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [15], [17], [18], [19]. However, the practicality of most of the proposed schemes is not quite clear.

The paper is structured as follows. After reviewing the case of decoding when transmitting over a channel with known channel law in Sec. II, we will deal with the case of unknown channel law in Sec. III. With the help of an excursion on the one-sweep algorithm in Sec. IV, we show in Sec. V that the one-sweep algorithm is very useful for the universal decoding setup. Finally, we conclude this paper in Sec. VI. Some more technical parts of the paper were relegated to the Appendix.

While being at MIT, the author was partially supported by NSF Grants CCF-0514801 and CCF-0515109 and by HP through the MIT/HP Alliance.

Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo Alto, CA 94304, USA. E-Mail: pascal.vontobel@ieee.org. The work for this paper was partially done while being at the Dept. of EECS, Massachusetts Institute of Technology, Cambridge, MA, USA.

This is essentially the paper that was published in the proceedings of the 44th Annual Allerton Conference on Communications, Control, and Computing, Sept. 27-29, 2006.

A. Notation

In the following, we will use Forney-style factor graphs (FFGs), also called normal factor graph [20], [21], [22]. A key feature of FFGs is that the variables are associated to the (half-)edges. Unless a variable is a measured quantity, we will use the capitalized variable name to label (half-)edges. All our codes will be binary, however, the results can easily be generalized to other alphabets. Vectors will usually be row vectors. We will use the abbreviation $\int_{\boldsymbol{\theta}}$ for $\int_{\mathcal{T}_{\boldsymbol{\Theta}}} \ldots \mathrm{d}\boldsymbol{\theta}$. Moreover, the expression $[A]$ is 1 if the statement A is true and 0 otherwise.

II. Decoding with Known Channel Law

In this section we briefly discuss how the factor-graph / message-passing iterative decoding framework [21], [20] can be used in the case where the channel law is known at the receiver. More precisely, we consider a DMC with input alphabet $\mathcal{X}$, output alphabet $\mathcal{Y}$, and channel law $W(y|x)$, i.e. $W(\mathbf{y}|\mathbf{x}) = \prod_{\ell=1}^{n} W(y_\ell|x_\ell)$. Assume that we are using a channel code $\mathcal{C}$ over $\mathcal{X}$ of length n and dimension k, that the sent codeword is denoted by the random vector $\mathbf{X}$ and that the received vector is denoted by the random vector $\mathbf{Y}$. Moreover, we assume that all codewords are sent equally likely, which means that $P_{\mathbf{X}}(\mathbf{x}) = 2^{-k} \cdot [\mathbf{x} \in \mathcal{C}]$ for all $\mathbf{x} \in \mathcal{X}^n$.

Two popular approaches for designing a decoder are to minimize to block error rate or the symbol error rate, respectively.

- If we minimize the block error rate then we obtain the so-called block-wise maximum a-posteriori (MAP) decision rule which says that upon observing the channel output $\mathbf{Y} = \mathbf{y}$ one decides

$$\begin{aligned}\hat{\mathbf{x}}^{\mathrm{block}}(\mathbf{y}) &= \arg\max_{\mathbf{x}\in\mathcal{X}^n} P_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y}) \\ &= \arg\max_{\mathbf{x}\in\mathcal{X}^n} P_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y}) \\ &= \arg\max_{\mathbf{x}\in\mathcal{X}^n} [\mathbf{x}\in\mathcal{C}] \cdot \prod_{\ell=1}^{n} P_{Y_\ell|X_\ell}(y_\ell|x_\ell) \\ &= \arg\max_{\mathbf{x}\in\mathcal{X}^n} [\mathbf{x}\in\mathcal{C}] \cdot \prod_{\ell=1}^{n} W(y_\ell|x_\ell).\end{aligned}$$

  For the components of $\hat{\mathbf{x}}^{\mathrm{block}}(\mathbf{y}) \triangleq \big(\hat{x}_1^{\mathrm{block}}(\mathbf{y}), \ldots, \hat{x}_n^{\mathrm{block}}(\mathbf{y})\big)$ this means that

$$\hat{x}_i^{\mathrm{block}}(\mathbf{y}) = \arg\max_{x_i\in\mathcal{X}} \max_{\substack{\mathbf{x}\in\mathcal{X}^n \\ x_i \text{ fixed}}} [\mathbf{x}\in\mathcal{C}] \cdot \prod_{\ell=1}^{n} W(y_\ell|x_\ell).$$

- If we minimize the symbol error rate then we obtain the so-called symbol-wise maximum a-posteriori decision

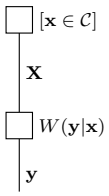

Fig. 1. Factor graph representing the factorization of $[\mathbf{x} \in \mathcal{C}] \cdot W(\mathbf{y}|\mathbf{x})$.

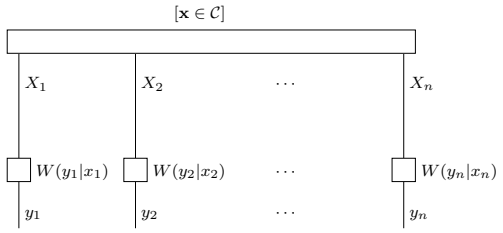

Fig. 2. Factor graph representing the factorization of $[\mathbf{x} \in \mathcal{C}] \cdot \prod_{\ell=1}^{n} W(y_\ell|x_\ell)$.

rule. Upon observing the channel output $\mathbf{Y} = \mathbf{y}$, it says that for all $i \in \{1, \ldots, n\}$ one decides

$$\begin{aligned}
\hat{x}_i^{\text{symb}}(\mathbf{y}) &= \arg\max_{x_i \in \mathcal{X}} P_{X_i|\mathbf{Y}}(x_i|\mathbf{y}) \\
&= \arg\max_{x_i \in \mathcal{X}} P_{X_i,\mathbf{Y}}(x_i, \mathbf{y}) \\
&= \arg\max_{x_i \in \mathcal{X}} \sum_{\substack{\mathbf{x} \in \mathcal{X}^n \\ x_i \text{ fixed}}} P_{\mathbf{X},\mathbf{Y}}(\mathbf{x}, \mathbf{y}) \\
&= \arg\max_{x_i \in \mathcal{X}} \sum_{\substack{\mathbf{x} \in \mathcal{X}^n \\ x_i \text{ fixed}}} [\mathbf{x} \in \mathcal{C}] \cdot \prod_{\ell=1}^{n} P_{Y_\ell|X_\ell}(y_\ell|x_\ell) \\
&= \arg\max_{x_i \in \mathcal{X}} \sum_{\substack{\mathbf{x} \in \mathcal{X}^n \\ x_i \text{ fixed}}} [\mathbf{x} \in \mathcal{C}] \cdot \prod_{\ell=1}^{n} W(y_\ell|x_\ell).
\end{aligned}$$

In both cases, we can use the FFG in Fig. 1 or the FFG in Fig. 2 to visualize the decoding. Because the FFGs are loopless, the decisions will be given by the max-product algorithm in the case of the block-wise MAP decision rule and by the sum-product algorithm in the case of the symbol-wise MAP decision rule.

The above decision rules yield only efficient algorithms if the message updates can be performed efficiently at the function nodes. This is e.g. the case when $\mathcal{C}$ is a convolutional code (with not too many states), a low-density parity-check (LDPC) code, or a turbo code. For the latter two cases, the function node representing the code indicator function $[\mathbf{x} \in \mathcal{C}]$ is replaced by an FFG that typically has loops. Note that in this case the max-product and the sum-product algorithms usually do not minimize, respectively, the block and symbol error rate. However, for suitably chosen LDPC and turbo codes very good performance results can be achieved with relatively low algorithmic complexity.

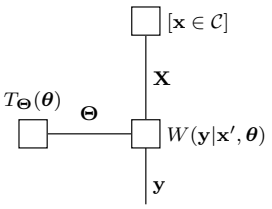

Fig. 3. FFG representing the factorization $[\mathbf{x} \in \mathcal{C}] \cdot T_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) \cdot W(\mathbf{y}|\mathbf{x}', \boldsymbol{\theta})$.

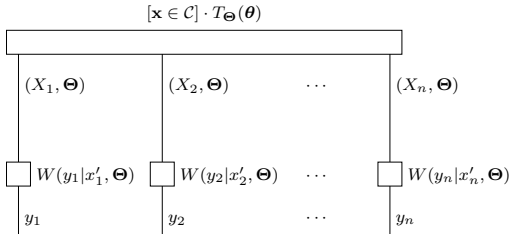

Fig. 4. FFG representing the factorization $[\mathbf{x} \in \mathcal{C}] \cdot T_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) \cdot \prod_{\ell=1}^{n} W(y_\ell|x_\ell', \boldsymbol{\theta})$.

III. Decoding with Unknown Channel Law: Universal Decoding

In contrast to Sec. II we now assume that we transmit over a DMC where the channel law is neither known by the sender nor by the receiver. Mathematically, we do this by replacing the channel law $W(y|x)$ by $W(y|x, \boldsymbol{\theta})$: the setup is such that the sender and the receiver know the input alphabet $\mathcal{X}$, the output alphabet $\mathcal{Y}$, the set $\mathcal{T}_{\boldsymbol{\Theta}}$ of possible $\boldsymbol{\theta}$-vectors, and the function $W(y|x, \boldsymbol{\theta})$, however they do not know the actual $\boldsymbol{\theta}$-vector. Subsequently, we will not directly transmit $\mathbf{x}$ but $\mathbf{x}'$ where $x_i' \triangleq \sigma_i(x_i)$ for some given permutations $\sigma_i(\,\cdot\,)$ of

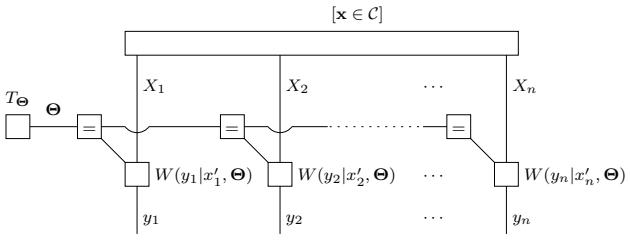

Fig. 5. Another FFG representing the factorization $[\mathbf{x} \in \mathcal{C}] \cdot T_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) \cdot \prod_{\ell=1}^{n} W(y_\ell|x_\ell', \boldsymbol{\theta})$

	block-wise MAP	symbol-wise MAP
with joint channel law estimate	①	③
without joint channel law estimate	②	④

TABLE I

Decision/estimation rules for the case where the channel law is unknown.

$\mathcal{X}$.[1]

Definition 1 *In the following, we assume that* $\boldsymbol{\theta} = (\theta_{x',y})_{(x',y)\in\mathcal{X}\times\mathcal{Y}}$*, that*

$$W(y|x',\boldsymbol{\theta}) \triangleq \theta_{x',y} \quad \textit{(for every } (x',y)\in\mathcal{X}\times\mathcal{Y}\textit{)},$$

and that

$$\mathcal{T}_{\boldsymbol{\Theta}} \triangleq \Big\{ \boldsymbol{\theta} \;\Big|\; \theta_{x',y} \geq 0 \textit{ for all } (x',y)\in\mathcal{X}\times\mathcal{Y}, \sum_y \theta_{x',y} = 1 \textit{ for all } x'\in\mathcal{X} \Big\}.$$

Moreover, we will take a Bayesian approach where the actual $\boldsymbol{\theta}$*-vector will be considered to be a realization of the random vector* $\boldsymbol{\Theta}$ *with a priori density* $T_{\boldsymbol{\Theta}}(\boldsymbol{\theta})$.[2] □

A popular way of obtaining low-complexity algorithms is to redraw an FFG, e.g. the FFG in Fig. 4 can be redrawn as the FFG in Fig. 5. Unfortunately, although the message updates would be rather simple, message-passing iterative decoding on the new FFG does not work: roughly speaking, the messages in the FFG will all be messages that favor none of the symbols in $\mathcal{X}$. The reason is that the algorithm works too locally and does not incorporate enough global information to break the symmetry.

So, in the following we will not split the factor node $[\mathbf{x}\in\mathcal{C}]\cdot T_{\boldsymbol{\Theta}}(\boldsymbol{\theta})$ into two factor nodes (as was done when going from Fig. 4 to Fig. 5). Our goal will therefore be to find algorithms that work efficiently despite the fact that we have "complicated" function nodes.

Paralleling the case of the known channel in Sec. II, we can derive a decision rule by minimizing either the block or the symbol error rate.[3] However, because not only $\mathbf{x}$ but also $\boldsymbol{\theta}$ is unknown, we can either estimate $\mathbf{x}$ by itself or jointly with $\boldsymbol{\theta}$. The resulting decision / estimation rules are shown in Table I and the formulas look as follows.

① Block-wise MAP decoding (with joint block-wise estimate of the channel parameters):

$$(\hat{\mathbf{x}},\hat{\boldsymbol{\theta}})(\mathbf{y}) = \arg\max_{(\mathbf{x},\boldsymbol{\theta})\in\mathcal{X}^n\times\mathcal{T}_{\boldsymbol{\Theta}}} [\mathbf{x}\in\mathcal{C}]\cdot T_{\boldsymbol{\Theta}}(\boldsymbol{\theta})\cdot\prod_{\ell=1}^n \theta_{x'_\ell,y_\ell}.$$

② Block-wise MAP decoding (without joint block-wise estimate of the channel parameters):

$$\hat{\mathbf{x}}(\mathbf{y}) = \arg\max_{\mathbf{x}\in\mathcal{X}^n} \int_{\boldsymbol{\theta}} [\mathbf{x}\in\mathcal{C}]\cdot T_{\boldsymbol{\Theta}}(\boldsymbol{\theta})\cdot\prod_{\ell=1}^n \theta_{x'_\ell,y_\ell}.$$

③ Symbol-wise MAP decoding (with joint block-wise estimate of the channel parameters):

$$(\hat{x}_i,\hat{\boldsymbol{\theta}})(\mathbf{y}) = \arg\max_{(x_i,\boldsymbol{\theta})\in\mathcal{X}\times\mathcal{T}_{\boldsymbol{\Theta}}} \sum_{\substack{\mathbf{x}\in\mathcal{X}^n\\ x_i \text{ fixed}}} [\mathbf{x}\in\mathcal{C}]\cdot T_{\boldsymbol{\Theta}}(\boldsymbol{\theta})\cdot\prod_{\ell=1}^n \theta_{x'_\ell,y_\ell}.$$

④ Symbol-wise MAP decoding (without joint block-wise estimate of the channel parameters):

$$\hat{x}_i(\mathbf{y}) = \arg\max_{x_i\in\mathcal{X}} \sum_{\substack{\mathbf{x}\in\mathcal{X}^n\\ x_i \text{ fixed}}} \int_{\boldsymbol{\theta}} [\mathbf{x}\in\mathcal{C}]\cdot T_{\boldsymbol{\Theta}}(\boldsymbol{\theta})\cdot\prod_{\ell=1}^n \theta_{x'_\ell,y_\ell}.$$

The resulting decision rules for the components of the above decision vectors are then

① Block-wise MAP decoding (with joint block-wise estimate of the channel parameters):

$$\hat{x}_i(\mathbf{y}) = \arg\max_{x_i\in\mathcal{X}} \max_{\substack{\mathbf{x}\in\mathcal{X}^n\\ x_i \text{ fixed}}} \max_{\boldsymbol{\theta}\in\mathcal{T}_{\boldsymbol{\Theta}}} [\mathbf{x}\in\mathcal{C}]\cdot T_{\boldsymbol{\Theta}}(\boldsymbol{\theta})\cdot\prod_{\ell=1}^n \theta_{x'_\ell,y_\ell}.$$

② Block-wise MAP decoding (without joint block-wise estimate of the channel parameters):

$$\hat{x}_i(\mathbf{y}) = \arg\max_{x_i\in\mathcal{X}} \max_{\substack{\mathbf{x}\in\mathcal{X}^n\\ x_i \text{ fixed}}} \int_{\boldsymbol{\theta}} [\mathbf{x}\in\mathcal{C}]\cdot T_{\boldsymbol{\Theta}}(\boldsymbol{\theta})\cdot\prod_{\ell=1}^n \theta_{x'_\ell,y_\ell}.$$

③ Symbol-wise MAP decoding (with joint block-wise estimate of the channel parameters):

$$\hat{x}_i(\mathbf{y}) = \arg\max_{x_i\in\mathcal{X}} \max_{\boldsymbol{\theta}\in\mathcal{T}_{\boldsymbol{\Theta}}} \sum_{\substack{\mathbf{x}\in\mathcal{X}^n\\ x_i \text{ fixed}}} [\mathbf{x}\in\mathcal{C}]\cdot T_{\boldsymbol{\Theta}}(\boldsymbol{\theta})\cdot\prod_{\ell=1}^n \theta_{x'_\ell,y_\ell}.$$

④ Symbol-wise MAP decoding (without joint block-wise estimate of the channel parameters):

$$\hat{x}_i(\mathbf{y}) = \arg\max_{x_i\in\mathcal{X}} \sum_{\substack{\mathbf{x}\in\mathcal{X}^n\\ x_i \text{ fixed}}} \int_{\boldsymbol{\theta}} [\mathbf{x}\in\mathcal{C}]\cdot T_{\boldsymbol{\Theta}}(\boldsymbol{\theta})\cdot\prod_{\ell=1}^n \theta_{x'_\ell,y_\ell}.$$

Let us point out the connection of some of the above decision rules to some universal decoders that were proposed so far in the literature. For this, we introduce $\big(a_{x',y}(\mathbf{x}',\mathbf{y})\big)_{(x',y)\in\mathcal{X}\times\mathcal{Y}}$ which counts the number of indices $\ell\in\{1,\ldots,n\}$ such that $x'_i = \sigma_i(x_i) = x'$ and $y_i = y$.

① Assuming $T_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) \propto 1$ for all $\boldsymbol{\theta}$, we can write

$$\begin{aligned}
&\hat{x}_i(\mathbf{y}) \\
&= \arg\max_{x_i\in\mathcal{X}} \max_{\substack{\mathbf{x}\in\mathcal{X}^n\\ x_i \text{ fixed}}} \max_{\boldsymbol{\theta}\in\mathcal{T}_{\boldsymbol{\Theta}}} [\mathbf{x}\in\mathcal{C}]\cdot T_{\boldsymbol{\Theta}}(\boldsymbol{\theta})\cdot\prod_{\ell=1}^n \theta_{x'_\ell,y_\ell} \\
&= \arg\max_{x_i\in\mathcal{X}} \max_{\substack{\mathbf{x}\in\mathcal{X}^n\\ x_i \text{ fixed}}} [\mathbf{x}\in\mathcal{C}]\cdot \max_{\boldsymbol{\theta}\in\mathcal{T}_{\boldsymbol{\Theta}}} T_{\boldsymbol{\Theta}}(\boldsymbol{\theta})\cdot\prod_{\ell=1}^n \theta_{x'_\ell,y_\ell} \\
&= \arg\max_{x_i\in\mathcal{X}} \max_{\substack{\mathbf{x}\in\mathcal{X}^n\\ x_i \text{ fixed}}} [\mathbf{x}\in\mathcal{C}]\cdot \max_{\boldsymbol{\theta}\in\mathcal{T}_{\boldsymbol{\Theta}}} \cdot \prod_{(x',y)\in\mathcal{X}\times\mathcal{Y}} \theta_{x',y}^{a_{x',y}(\mathbf{x}',\mathbf{y})}.
\end{aligned}$$

We see that the maximizing $\boldsymbol{\theta}^*$ is $\theta^*_{x',y} = a_{x',y}(\mathbf{x}',\mathbf{y})/\sum_{\tilde{y}\in\mathcal{Y}} a_{x',\tilde{y}}(\mathbf{x}',\mathbf{y})$ for all $(x',y)\in\mathcal{X}\times\mathcal{Y}$. We obtain $\prod_{(x',y)\in\mathcal{X}\times\mathcal{Y}} (\theta^*_{x',y})^{a_{x',y}(\mathbf{x}',\mathbf{y})} =$

[1] We will not discuss much further why we introduce the permutations $\sigma_i(\cdot)$. The main reason is that we would like to use a linear code $\mathcal{C}$. Permuting the symbols helps insure that the zero codeword does not have a special position in the codebook. Moreover, if $\mathbf{1}$ is in the codebook, then for every $\mathbf{x}\in\mathcal{C}$, $\mathbf{x}+\mathbf{1}$ (mod 2) is in $\mathcal{C}$ and this symmetry complicates universal decoding if it is not broken by the permutations $\sigma_i(\cdot)$.

[2] For computational reasons it is more important that $T_{\boldsymbol{\Theta}}(\boldsymbol{\theta})$ has a convenient form than that it exactly represents the a priori density of $\boldsymbol{\Theta}$ (if such a density exists at all). As long as no parameter setting is excluded a priori, it can be seen from the expressions later in this paper that for large block length n the influence of the a priori distribution is negligible.

[3] It seems that in the past, people have mainly focused on minimizing the word error rate in the context of universal channel decoding.

$\exp\big(-nH(\mathbf{y}|\mathbf{x})\big)$, where $H(\mathbf{y}|\mathbf{x})$ is the empirical conditional entropy of $\mathbf{y}$ given $\mathbf{x}$. Letting $H(\mathbf{y})$ be the empirical entropy of $\mathbf{y}$ (note that it is only a function of the observed channel output $\mathbf{y}$ and not of $\mathbf{x}$, and letting $I(\mathbf{x};\mathbf{y}) = H(\mathbf{y}) - H(\mathbf{y}|\mathbf{x})$ be the empirical mutual information of $\mathbf{x}$ and $\mathbf{y}$, we get

$$\hat{x}_i(\mathbf{y}) = \arg\max_{x_i\in\mathcal{X}} \max_{\substack{\mathbf{x}\in\mathcal{X}^n \\ x_i \text{ fixed}}} [\mathbf{x}\in\mathcal{C}]\cdot\exp\big(nI(\mathbf{x};\mathbf{y})\big).$$

This is the maximum mutual information (MMI) decoder of Csiszár and Körner [9]. Recently, Coleman et al. [18], [19] have studied relaxation approaches to this decision rule where $[\mathbf{x}\in\mathcal{C}]$ was relaxed according to the recipes in [23], [24]. (Actually, Coleman et al. studied not the universal channel coding setup but rather the "dual" universal source coding setup.)

② Setting $T_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) \propto \prod_{(x',y)\in\mathcal{X}\times\mathcal{Y}} \theta_{x',y}^{-1+1/2} = \theta_{x',y}^{-1/2}$ for all $\boldsymbol{\theta}$ (which corresponds to Dirichlet-$(1/2,\ldots,1/2)$ distributions for each $x'\in\mathcal{X}$), we can write

$$\begin{aligned}
&\hat{x}_i(\mathbf{y}) \\
&= \arg\max_{x_i\in\mathcal{X}} \max_{\substack{\mathbf{x}\in\mathcal{X}^n \\ x_i \text{ fixed}}} \int_{\boldsymbol{\theta}} [\mathbf{x}\in\mathcal{C}]\cdot T_{\boldsymbol{\Theta}}(\boldsymbol{\theta})\cdot\prod_{\ell=1}^{n}\theta_{x'_\ell,y_\ell} \\
&= \arg\max_{x_i\in\mathcal{X}} \max_{\substack{\mathbf{x}\in\mathcal{X}^n \\ x_i \text{ fixed}}} [\mathbf{x}\in\mathcal{C}]\cdot\int_{\boldsymbol{\theta}} \prod_{(x',y)\in\mathcal{X}\times\mathcal{Y}} \theta_{x',y}^{a_{x',y}(\mathbf{x}',\mathbf{y})-1/2} \\
&= \arg\min_{x_i\in\mathcal{X}} \min_{\substack{\mathbf{x}\in\mathcal{X}^n \\ x_i \text{ fixed}}} [\mathbf{x}\in\mathcal{C}]\cdot\exp\big(l_{\mathrm{KT}}(\mathbf{y}|\mathbf{x}')\big),
\end{aligned}$$

where $l_{\mathrm{KT}}(\mathbf{y}|\mathbf{x}')$ is the compression length when compressing $\mathbf{y}$ given $\mathbf{x}'$ according to the Krichevsky-Trofimov universal measure [25]. The resulting decoding rule is very similar to the decoding rule in [14, Sec. V] where $l_{\mathrm{KT}}(\mathbf{x}'|\mathbf{y})$ is minimized over the codewords. Instead of using the source compression length according to the Krichevsky-Trofimov universal measure, one can also take the source compression length that is obtained by applying the Lempel-Ziv algorithm, see e.g. [10], [13].

In all these decision / estimation setups considered above, the expression for $\hat{x}_i(\mathbf{y})$ has the same structure. Starting from the right-hand side and going to the left-hand side, the expression contains the product $[\mathbf{x}\in\mathcal{C}]\cdot T_{\boldsymbol{\Theta}}(\boldsymbol{\theta})\cdot\prod_{\ell=1}^{n}\theta_{x'_\ell,y_\ell}$ (which in the factor-graph framework is represented by an FFG), then there is the "marginalization" operation, and finally $\arg\max_{x_i\in\mathcal{X}}$ chooses the best x_i.

Let us now have a closer look at the "marginalization" operation for the various cases.

① In this case, the "marginalization" corresponds to the max-product algorithm. Note though that $\boldsymbol{\theta}$ is continuous and so it might not be easy to perform the maximization over $\boldsymbol{\theta}$. It seems to be better to leave this maximization untouched until $\max_{\substack{\mathbf{x}\in\mathcal{X}^n \\ x_i \text{ fixed}}}$ is performed.

② Algebraically, we have three different operations here: maximization, summation (we consider integration as a summation), and the product of non-negative real numbers. This is obviously not a semi-ring as is required for the factor-graph / message-passing iterative decoding framework.

③ Similar to the above case, here we have three different operations: maximization, summation, and the product of non-negative real numbers.

④ In this case, the "marginalization" corresponds to the sum-product algorithm.

We see that in the cases ② and ③ we do not have a semi-ring, in the case ① we have a semi-ring that is not a ring (there is no "additive inverse"), and in the case ④ we have a semi-ring that is also a ring. It seems therefore that the case ④ (symbol-wise MAP decoding without joint block-wise estimate of the channel parameters) has more algebraic structure than the other decision rules. It is certainly desirable to use this additional structure to perform the message updates more efficiently. We will use the one-sweep algorithm to make a case for this claim; To that end, the next section gives a review of this algorithm. Afterwards, we will discuss its use in the universal setup.

IV. The One-Sweep Algorithm

Assume to have a (partial) FFG as in Fig. 6, where the function node represents the indicator function of a linear (sub)code $\mathcal{C}$. During an update of this function node we compute the outgoing messages $\mu_{f\to X_i}(x_i)$, $i=1,\ldots,n$, based on the incoming messages $\mu_{X_i\to f}(x_i)$, $i=1,\ldots,n$, using the sum-product algorithm. In the following, we will assume that the underlying semi-ring is a ring, i.e. that every element of the ring has an additive inverse.

If the subcode is represented by a trellis, one efficient way to solve this update task is by using the BCJR algorithm [26], or equivalently, the forward-backward algorithm.

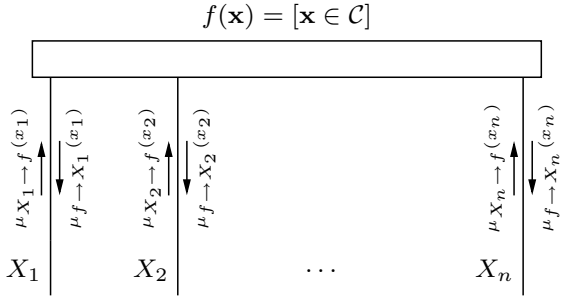

Fig. 6. (Partial) FFG where the function node $f(\mathbf{x}) = [\mathbf{x}\in\mathcal{C}]$ represents the subcode $\mathcal{C}$.

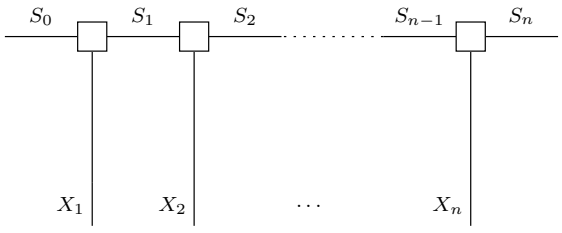

Fig. 7. Detailed version of the partial factor graph shown in Fig. 6.

However, there are alternatives when the underlying semiring is actually a ring. One of them is to do the computations using the dual code [27] (see also [20]). Another very interesting alternative is the one-sweep algorithm by Johansson and Zigangirov [28]. In contrast to the BCJR algorithm, which needs a forward and a backward sweep, the one-sweep algorithm only needs a slightly more complex forward sweep with some simple processing at the end.[4]

In [29, App. D] we have already discussed the relationship between these three algorithms that perform the above-mentioned message update, especially we showed a simplified derivation of the one-sweep algorithm. Here we will discuss an even more simplified derivation of this algorithm. Originally, the one-sweep algorithm was presented as an algorithm for computing the a-posteriori probabilities when decoding a code. However, in our opinion, the slightly more abstract setup of message updates helps considerably in obtaining a rather straightforward derivation of the algorithm.

In the following, we will assume that $\mathcal{C}$ is a binary linear code of length n defined by a parity-check matrix $\mathbf{H}$. (The results here can easily be generalized to linear codes over other finite fields.) Let $\mathcal{I} \triangleq \{1, \ldots, n\}$. To simplify notation, we will use for all $i \in \mathcal{I}$ the abbreviation $\mu_i(x_i) \triangleq \mu_{X_i \to f}(x_i)$ for the incoming messages. With this, the outgoing message $\mu_{f \to X_i}(x_i)$, $i \in \mathcal{I}$, is given by

$$\mu_{f \to X_i}(0) \triangleq \sum_{\substack{\mathbf{x} \in \mathcal{X}^n \\ x_i = 0}} [\mathbf{x} \in \mathcal{C}] \prod_{\substack{\ell \in \mathcal{I} \\ \ell \neq i}} \mu_\ell(x_\ell) = \sum_{\substack{\mathbf{x} \in \mathcal{C} \\ x_i = 0}} \prod_{\substack{\ell \in \mathcal{I} \\ \ell \neq i}} \mu_\ell(x_\ell),$$

$$\mu_{f \to X_i}(1) \triangleq \sum_{\substack{\mathbf{x} \in \mathcal{X}^n \\ x_i = 1}} [\mathbf{x} \in \mathcal{C}] \prod_{\substack{\ell \in \mathcal{I} \\ \ell \neq i}} \mu_\ell(x_\ell) = \sum_{\substack{\mathbf{x} \in \mathcal{C} \\ x_i = 1}} \prod_{\substack{\ell \in \mathcal{I} \\ \ell \neq i}} \mu_\ell(x_\ell).$$

Let $\mathbf{e}_i$ be the all-zeros row vector of length n except for a "1" at position i and let $\mathcal{C}_i \triangleq \mathcal{C} + \mathbf{e}_i$ ($i \in \mathcal{I}$) be cosets of the code $\mathcal{C}$.

The main idea of the one-sweep algorithm is to compute the intermediate quantities χ, χ_1, …, χ_n from which the outgoing messages can be obtained efficiently. If χ, χ_1, …, χ_n can be computed efficiently, then also the outgoing messages can be computed efficiently.

Definition 2 *We define*

$$\chi \triangleq \sum_{\mathbf{x} \in \mathcal{C}} \prod_{\ell=1}^{n} \mu_\ell(x_\ell),$$

$$\chi_i \triangleq \sum_{\mathbf{x} \in \mathcal{C}_i} \prod_{\ell=1}^{n} \mu_\ell(x_\ell) \quad \textit{(for all } i \in \mathcal{I}\textit{)}.$$

□

Theorem 3 *Let $i \in \mathcal{I}$. If $\Delta_i \triangleq \mu_i(0)^2 - \mu_i(1)^2$ is a multiplicatively invertible element of the underlying ring, the i-th outgoing message $\mu_{f \to X_i}(x_i)$ is given by*

$$\begin{pmatrix} \mu_{f \to X_i}(0) \\ \mu_{f \to X_i}(1) \end{pmatrix} = \frac{1}{\Delta_i} \begin{pmatrix} +\mu_i(0) & -\mu_i(1) \\ -\mu_i(1) & +\mu_i(0) \end{pmatrix} \cdot \begin{pmatrix} \chi \\ \chi_i \end{pmatrix},$$

where χ and χ_i were defined in Def. 2.

Proof: See App. B. □

In the case that for some $i \in \mathcal{I}$ the expression $\Delta_i = \mu_i(0)^2 - \mu_i(1)^2$ is not a multiplicatively invertible element, one has to use different procedures or one has to use some reformulation of the setup (e.g. one can reorder the code bit positions and use a modified one-sweep algorithm, see the remarks after Eq. (24) in [28]).

In App. A we show how the quantities χ, χ_1, …, χ_n in Def. 2 can be computed efficiently. From that discussion it will become clear that the name "one-sweep algorithm" is indeed justified.

V. The One-Sweep Algorithm for Universal Decoding

Let us briefly discuss the application of the one-sweep algorithm to the update of the function node $[\mathbf{x} \in \mathcal{C}] \cdot T_{\boldsymbol{\Theta}}(\boldsymbol{\theta})$ in Fig. 4. Here, the i-th incoming message is

$$\mu_i(0) = \gamma_{0,i} \cdot \theta_{\sigma_i(0), y_i},$$
$$\mu_i(1) = \gamma_{1,i} \cdot \theta_{\sigma_i(1), y_i}$$

for some non-negative real values $\gamma_{0,i}$ and $\gamma_{1,i}$. We observe that the computations for obtaining χ, χ_1, …, χ_n involve only the following computations:

- we only multiply polynomials by monomials;
- we add two polynomials.

However, we do *not* need to compute the product of two general polynomials.

Moreover, the division by

$$\begin{aligned} \Delta_i &= \mu_i(0)^2 - \mu_i(1)^2 \\ &= \gamma_{0,i}^2 \cdot \theta_{\sigma_i(0), y_i}^2 - \gamma_{1,i}^2 \cdot \theta_{\sigma_i(1), y_i}^2 \\ &= +\gamma_{0,i}^2 \theta_{\sigma_i(0), y_i}^2 \left(1 - \frac{\gamma_{1,i}^2}{\gamma_{0,i}^2} \cdot \frac{\theta_{\sigma_i(1), y_i}^2}{\theta_{\sigma_i(0), y_i}^2} \right) \\ &= -\gamma_{1,i}^2 \theta_{\sigma_i(1), y_i}^2 \left(1 - \frac{\gamma_{0,i}^2}{\gamma_{1,i}^2} \cdot \frac{\theta_{\sigma_i(0), y_i}^2}{\theta_{\sigma_i(1), y_i}^2} \right) \end{aligned}$$

can be preformed quite efficiently as a long division. Depending if the ratio $\gamma_{0,i}/\gamma_{1,i}$ is larger or smaller than one, it is advisable to use one or the other representation of Δ_i.

Note that when performing the BCJR algorithm, the forward and the backward recursions will also only involve the multiplication of polynomials by monomials and the addition of two monomials. However, when computing the outgoing messages based on the forward and backward messages, one needs to compute the product of two polynomials. It might be possible to compute this product (which can be seen as a multidimensional convolution) via the Fourier domain, however, given the dynamical range of the coefficients, it is not clear how robust this approach is.

In some contexts the approach of computing the outgoing messages via the dual code might also be an efficient alternative. However, some considerations (note e.g. that the i-th dual incoming message is $(\gamma_{0,i} \cdot \theta_{\sigma_i(0), y_i} + \gamma_{1,i} \cdot \theta_{\sigma_i(1), y_i}, \gamma_{0,i} \cdot \theta_{\sigma_i(0), y_i} - \gamma_{1,i} \cdot \theta_{\sigma_i(1), y_i})$, i.e. both components

[4] The one-sweep algorithm is non-local in the sense that the mentioned processing at the end is non-local in the factor graph shown in Fig. 7.

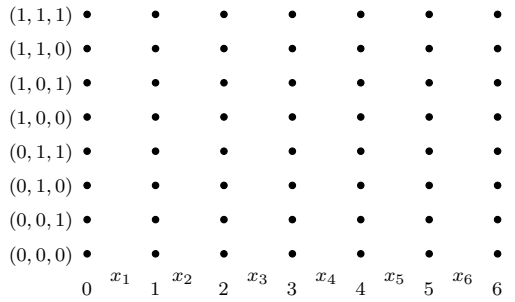

Fig. 8. Grid for a trellis of a code having a parity-check matrix with $n = 6$ columns and $r = 3$ rows.

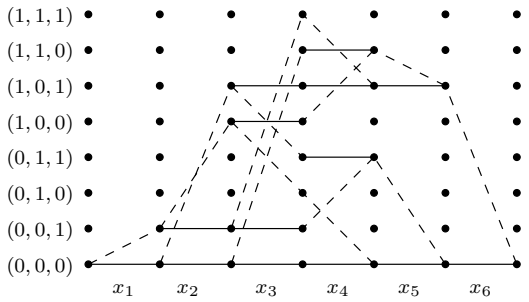

Fig. 9. Trellis for Ex. 6 (Solid line: "0", dashed line: "1").

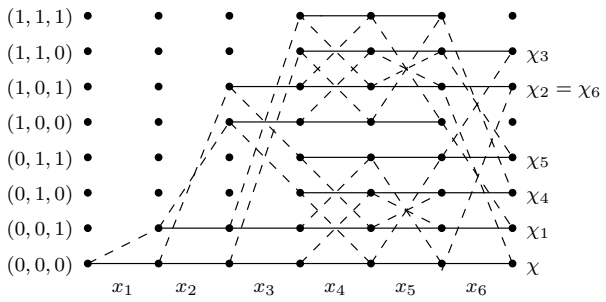

Fig. 10. Extended trellis for Ex. 6 (Solid line: "0", dashed line: "1").

are *not* monomials) show that it is not quite clear that this approach will be more efficient than the one-sweep algorithm or the BCJR algorithm in general (but perhaps it is in some special cases).

VI. Conclusions and Outlook

In this paper we have made some initial considerations within the factor-graph / message-passing iterative decoding framework concerning efficient algorithms for universal channel decoding. We envision that in order to construct well-performing and efficient universal decoders, large codes will have to be built out of subcodes where the message updates of the subcodes will be done according to the algorithms that we have outlined above. It will also be interesting to see how the knowledge about the channel parameters can be efficiently gathered from the subcodes and used in further iterations.

VII. Acknowledgments

We are indebted to Amos Lapidoth for pointing out to us the very interesting subject of universal channel decoding and for asking if the insights from [30] can be used in the context of universal channel decoding.

Appendix

A. Construction of Trellises and Extended Trellises

In this appendix we discuss a method by Bahl et al. [26] for constructing an optimal[5] trellis for a binary linear $[n, k]$ code. They assume that the code is defined by a parity-check matrix like

$$\mathbf{H} \triangleq \begin{bmatrix} \mathbf{h}_1^\mathsf{T} & \mathbf{h}_2^\mathsf{T} & \cdots & \mathbf{h}_n^\mathsf{T} \end{bmatrix}.$$

This trellis construction method by Bahl et al. is also the one used in [28]. Let r be the number of rows of $\mathbf{H}$. For simplicity in exposition, we will make in the following the reasonable assumption that $\mathbf{h}_i^\mathsf{T} \neq \mathbf{0}^\mathsf{T}$ for all $i \in \mathcal{I}$.

Definition 4 *Let $i \in \mathcal{I}$.*

- *If $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is a row vector of length n, we let $\mathbf{x}_{[1:i]} \triangleq (x_1, x_2, \ldots, x_i)$.*
- *Let $\mathbf{H}_{[1:i]} \triangleq \left[\mathbf{h}_1^\mathsf{T}, \mathbf{h}_2^\mathsf{T}, \ldots, \mathbf{h}_i^\mathsf{T}\right]$ be a submatrix of $\mathbf{H}$.*
- *Let $\mathbf{s}_0 \triangleq \mathbf{0}$ and let the row vectors $\mathbf{s}_i = \mathbf{x}_{[1:i]} \cdot \mathbf{H}_{[1:i]}^\mathsf{T}$ of length r be the so-called partial syndromes.*
- *In a trellis each path from left to right represents a codeword and all paths should start and end in the same state (usually the zero state). Moreover, for each codeword there is such a path.*

□

Algorithm 5 (to construct a trellis for a block code) *Given a block code, a trellis is now drawn according to the following steps.*

- *Draw a grid of nodes with 2^r rows and $n+1$ columns and do the labeling at the bottom and on the left-hand side of the grid as shown in Fig. 8. The vertical labels are the different possible partial syndromes, i.e. all binary vectors of length r. The horizontal positions are labeled from 0 to n whereas the label of code-bit x_i is in-between position $i-1$ and i, $i \in \mathcal{I}$.*
- *The paths through the trellis are constructed in the following way. For each $\mathbf{x} \in \mathcal{C}$ one calculates the partial syndromes $\mathbf{s}_i$ for all $i \in \{0, 1, \ldots, n\}$; the path of $\mathbf{x}$ through the trellis is then the unique path that goes through state $\mathbf{s}_i$ at position i for all $i \in \{0, 1, \ldots, n\}$. If the path between position $i-1$ and position i connects two different partial syndromes the label is "1", else it is "0". Note that $\mathbf{s}_0 = \mathbf{0}$ and $\mathbf{s}_n = \mathbf{0}$ for all $\mathbf{x} \in \mathcal{C}$, so all the paths start and end in the same state (as it should be by definition of a trellis).*

Example 6 (First trellis construction example) As a first example we consider the binary linear $[6, 3, 2]$ code with the

[5]By optimal we mean optimal in various senses, see e.g. [31]. For other orderings of the time axis there might be realizations with less states.

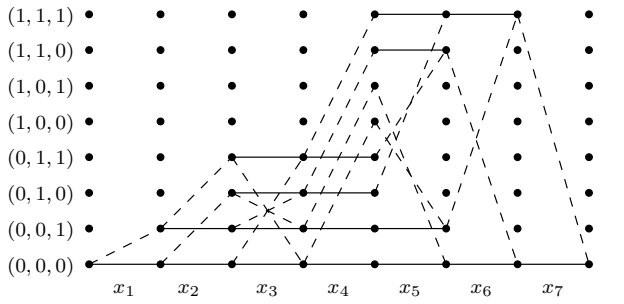

Fig. 11. Trellis for Ex. 7 (Solid line: "0", dashed line: "1").

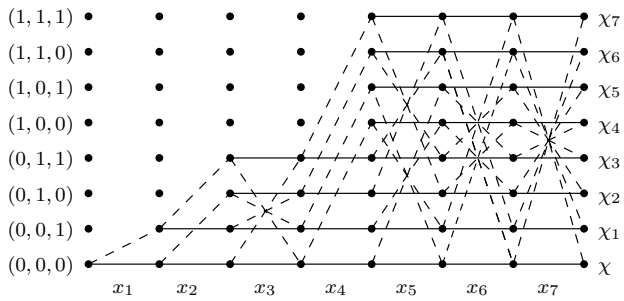

Fig. 12. Extended trellises for Ex. 7 (Solid line: "0", dashed line: "1").

parity-check matrix

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

Performing Alg. 5 for the given code results in the trellis shown in Fig. 9. E.g., for the codeword $\mathbf{x} = (1, 1, 0, 1, 1, 1)$ we get the partial syndromes

$$\mathbf{s}_0 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}, \ \mathbf{s}_1 = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}, \ \mathbf{s}_2 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix},$$
$$\mathbf{s}_3 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}, \ \mathbf{s}_4 = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}, \ \mathbf{s}_5 = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix},$$
$$\mathbf{s}_6 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}.$$

□

Example 7 (Second trellis construction example) As a second example we consider the binary $[7, 4, 3]$ Hamming code given by the parity-check matrix

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Performing Alg. 5 we get the trellis shown in Fig. 11. □

We extend the above trellis construction in Alg. 5 in the following way: we not only draw all paths for $\mathbf{x} \in \mathcal{C}$, but also the paths associated to all $\mathbf{x} \in \mathcal{C}_i$ for all $i \in \mathcal{I}$. By this extension we get for the codes in Exs. 6 and 7 the extended trellises in Fig. 10 and Fig. 12, respectively. The labels χ_i on the right-hand side are attached to the level where all codewords of a coset end.[6] The label χ is attached to the level $\mathbf{0}$ where all paths of the codewords of $\mathcal{C}$ end. Note that not all possible final syndromes are associated with a χ or χ_i; however, there are some which are connected to several χ_i's, namely exactly when there are several identical columns in the parity-check matrix.

[6]Note that $\mathbf{s}_n = \mathbf{x} \cdot \mathbf{H}^{\mathsf{T}} = \mathbf{e}_i \cdot \mathbf{H}^{\mathsf{T}} = \mathbf{h}_i$ is the same for all vectors $\mathbf{x}$ in the same coset $\mathcal{C}_i$.

Remark 8 (Special property of Ex. 7) The parity-check matrix in Ex. 7 is special in the sense that all possible columns except the zero column appear exactly once. Thus to every final syndrome we can associate exactly a $\chi, \chi_1, \ldots, \chi_n$. (Note, $n = 2^{n-k} - 1$, where k is the dimension and $n - k$ the redundancy of the code, respectively.) □

Algorithm 9 (to calculate χ and χ_i $(i \in \mathcal{I})$)

- *Set*

$$\chi(\mathbf{s}, 0) \triangleq \begin{cases} 1 & (\mathbf{s} = \mathbf{0}), \\ 0 & (\mathbf{s} \neq \mathbf{0}), \end{cases}.$$

  for all binary vectors $\mathbf{s}$ *of length* r.
- *Given* $\mu_i(x_i) = \mu_{X_i \to f}(x_i)$*, perform the following calculations for all* i *from* 1 *to* n *for all binary vectors* $\mathbf{s}$ *of length* r.

$$\chi(\mathbf{s}, i) \triangleq \chi(\mathbf{s}, i-1)\mu_i(0) + \chi(\mathbf{s} - \mathbf{h}_i, i-1)\mu_i(1).$$

  (More specifically, this has only to be done for the states $\mathbf{s}$ *at position* i *of the extended trellis if there is at least one branch incident from the left.)*
- *Finally, we set*

$$\chi \triangleq \chi(\mathbf{0}, n) \quad \textit{and} \quad \chi_i \triangleq \chi(\mathbf{h}_i, n) \quad \textit{(for all } i \in \mathcal{I}\textit{)}.$$

Lemma 10 *Algorithm 9 calculates the quantities* χ*,* χ_1*, ...,* χ_n *as defined in Def. 2.*

Proof: We leave it to the reader to check the validity of Algorithm 9. □

The above algorithm is of course akin to the forward recursion of the BCJR algorithm, but it is now performed on the extended trellis instead of on the original trellis.

B. Proof of Theorem 3

Proof: We will first show that

$$\begin{pmatrix} \mu_i(0) & \mu_i(1) \\ \mu_i(1) & \mu_i(0) \end{pmatrix} \cdot \begin{pmatrix} \mu_{f \to X_i}(0) \\ \mu_{f \to X_i}(1) \end{pmatrix} = \begin{pmatrix} \chi \\ \chi_i \end{pmatrix}. \tag{1}$$

Indeed, the first line in (1) follows from

$$\begin{aligned} &\mu_i(0) \cdot \mu_{f \to X_i}(0) + \mu_i(1) \cdot \mu_{f \to X_i}(1) \\ &\quad = \mu_i(0) \sum_{\substack{\mathbf{x} \in \mathcal{C} \\ x_i = 0}} \prod_{\substack{\ell \in \mathcal{I} \\ \ell \neq i}} \mu_\ell(x_\ell) + \mu_i(1) \sum_{\substack{\mathbf{x} \in \mathcal{C} \\ x_i = 1}} \prod_{\substack{\ell \in \mathcal{I} \\ \ell \neq i}} \mu_\ell(x_\ell) \\ &\quad = \sum_{\substack{\mathbf{x} \in \mathcal{C} \\ x_i = 0}} \prod_{\ell \in \mathcal{I}} \mu_\ell(x_\ell) + \sum_{\substack{\mathbf{x} \in \mathcal{C} \\ x_i = 1}} \prod_{\ell \in \mathcal{I}} \mu_\ell(x_\ell) \\ &\quad = \sum_{\mathbf{x} \in \mathcal{C}} \prod_{\ell \in \mathcal{I}} \mu_\ell(x_\ell) = \chi, \end{aligned}$$

and the second line in (1) follows from

$$\begin{aligned}
&\mu_i(1)\cdot\mu_{f\to X_i}(0)+\mu_i(0)\cdot\mu_{f\to X_i}(1)\\
&\quad=\mu_i(1)\sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=0}}\prod_{\substack{\ell\in\mathcal{I}\\ \ell\neq i}}\mu_\ell(x_\ell)+\mu_i(0)\sum_{\substack{\mathbf{x}\in\mathcal{C}\\ x_i=1}}\prod_{\substack{\ell\in\mathcal{I}\\ \ell\neq i}}\mu_\ell(x_\ell)\\
&\quad\overset{(*)}{=}\mu_i(1)\sum_{\substack{\tilde{\mathbf{x}}\in\mathcal{C}_i\\ \tilde{x}_i=1}}\prod_{\substack{\ell\in\mathcal{I}\\ \ell\neq i}}\mu_\ell(\tilde{x}_\ell)+\mu_i(0)\sum_{\substack{\tilde{\mathbf{x}}\in\mathcal{C}_i\\ \tilde{x}_i=0}}\prod_{\substack{\ell\in\mathcal{I}\\ \ell\neq i}}\mu_\ell(\tilde{x}_\ell)\\
&\quad=\sum_{\substack{\tilde{\mathbf{x}}\in\mathcal{C}_i\\ \tilde{x}_i=1}}\prod_{\ell\in\mathcal{I}}\mu_\ell(\tilde{x}_\ell)+\sum_{\substack{\tilde{\mathbf{x}}\in\mathcal{C}_i\\ \tilde{x}_i=0}}\prod_{\ell\in\mathcal{I}}\mu_\ell(\tilde{x}_\ell)\\
&\quad=\sum_{\tilde{\mathbf{x}}\in\mathcal{C}_i}\prod_{\ell\in\mathcal{I}}\mu_\ell(\tilde{x}_\ell)=\chi_i.
\end{aligned}$$

Note that in step $(*)$ we replaced every summand $\mathbf{x}\in\mathcal{C}$ by $\tilde{\mathbf{x}}\triangleq\mathbf{x}+\mathbf{e}_i\in\mathcal{C}_i$ and used the fact that $\prod_{\ell\in\mathcal{I},\,\ell\neq i}\mu_\ell(x_\ell)=\prod_{\ell\in\mathcal{I},\,\ell\neq i}\mu_\ell(\tilde{x}_\ell)$.

Solving the linear equation system in (1) yields the result in Th. 3. □

REFERENCES

[1] A. P. Worthen and W. E. Stark, "Unified design of iterative receivers using factor graphs," *IEEE Trans. on Inform. Theory*, vol. IT–48, no. 2, pp. 843–849, Feb. 2001.

[2] R. Nuriyev and A. Anastasopoulos, "Pilot-symbol-assisted coded transmission over the block-noncoherent AWGN channel," *IEEE Trans. on Comm.*, vol. COM–51, no. 6, pp. 953–963, June 2003.

[3] H. Steendam, N. Noels, and M. Moeneclaey, "Iterative carrier phase synchronization for low-density parity-check coded systems," in *Proc. IEEE Intern. Conf. Communications*, vol. 5, Anchorage, AK, USA, May 11–15 2003, pp. 3120–3124.

[4] J. Dauwels and H.-A. Loeliger, "Joint decoding and phase estimation: an exercise in factor graphs," in *Proc. IEEE Intern. Symp. on Inform. Theory*, Pacifico Yokohama, Japan, June 29 – July 4 2003, p. 231.

[5] ——, "Phase estimation by message passing," in *Proc. IEEE Intern. Conf. Communications*, vol. 1, Paris, France, June 20–24 2004, pp. 523–527.

[6] J. Dauwels, S. Korl, and H.-A. Loeliger, "Expectation maximization for phase estimation," in *Proc. Eighth Intern. Symp. on Comm. Theory and Appl.*, Ambleside, England, 2005.

[7] V. D. Goppa, "Universal decoding for symmetric channels," *Probl. Inform. Transm.*, vol. 11, no. 1, pp. 15–22, 1975.

[8] ——, "Nonprobabilistic mutual information without memory," *Probl. Contr. Inform. Theory*, vol. 4, pp. 97–102, 1975.

[9] I. Csiszár and J. Körner, *Information Theory*. Budapest: Akadémiai Kiadó (Publishing House of the Hungarian Academy of Sciences), 1981, coding theorems for discrete memoryless systems.

[10] J. Ziv, "Universal decoding for finite-state channels," *IEEE Trans. on Inform. Theory*, vol. IT–31, no. 4, pp. 453–460, July 1985.

[11] N. Merhav, "Universal decoding for memoryless Gaussian channels with a deterministic interference," *IEEE Trans. on Inform. Theory*, vol. IT–39, no. 4, pp. 1261–1269, July 1993.

[12] M. Feder and A. Lapidoth, "Universal decoding for channels with memory," *IEEE Trans. on Inform. Theory*, vol. IT–44, no. 5, pp. 1726–1745, Sept. 1998.

[13] A. Lapidoth and J. Ziv, "On the universality of the LZ-based decoding algorithm," *IEEE Trans. on Inform. Theory*, vol. IT–44, no. 5, pp. 1746–1755, Sept. 1998.

[14] ——, "On the decoding of convolutional codes on an unknown channel," *IEEE Trans. on Inform. Theory*, vol. IT–45, no. 7, pp. 2321–2332, Nov. 1999.

[15] M. Feder and N. Merhav, "Universal composite hypothesis testing: a competitive minimax approach," *IEEE Trans. on Inform. Theory*, vol. IT–48, no. 6, pp. 1504–1517, June 2002.

[16] Y. Ephraim and N. Merhav, "Hidden Markov processes," *IEEE Trans. on Inform. Theory*, vol. 48, no. 6, pp. 1518–1569, June 2002.

[17] A. Lapidoth and P. Narayan, "Reliable communication under channel uncertainty," *IEEE Trans. on Inform. Theory*, vol. IT–44, no. 6, pp. 2148–2177, Oct. 1998.

[18] T. P. Coleman, M. Médard, and M. Effros, "Linear complexity universal decoding with exponential error probability decay," in *Proc. 2005 International Conference on Wireless Networks, Communications, and Mobile Computing (Wirelesscom 2005)*, Maui, HI, USA, Jun. 13-16 2005.

[19] T. P. Coleman and M. Médard, "On low complexity decodable universally good linear codes," in *Proc. Inaugural Workshop of the Center for Information Theory and its Applications*, UC San Diego, La Jolla, CA, USA, Feb. 6-10 2006.

[20] G. D. Forney, Jr., "Codes on graphs: normal realizations," *IEEE Trans. on Inform. Theory*, vol. IT–47, no. 2, pp. 520–548, Feb. 2001.

[21] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. on Inform. Theory*, vol. IT–47, no. 2, pp. 498–519, Feb. 2001.

[22] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Sig. Proc. Mag.*, vol. 21, no. 1, pp. 28–41, Jan. 2004.

[23] J. Feldman, "Decoding error-correcting codes via linear programming," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 2003.

[24] J. Feldman, M. J. Wainwright, and D. R. Karger, "Using linear programming to decode binary linear codes," *IEEE Trans. on Inform. Theory*, vol. IT–51, no. 3, pp. 954–972, May 2005.

[25] R. E. Krichevsky and V. K. Trofimov, "The performance of universal encoding," *IEEE Trans. on Inform. Theory*, vol. IT–27, no. 2, pp. 199–207, Mar. 1981.

[26] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. on Inform. Theory*, vol. IT–20, no. 2, pp. 284–287, Mar. 1974.

[27] C. R. P. Hartmann and L. D. Rudolph, "An optimum symbol-by-symbol decoding rule for linear codes," *IEEE Trans. on Inform. Theory*, vol. IT–22, no. 5, pp. 514–517, 1976.

[28] T. Johansson and K. Zigangirov, "A simple one-sweep algorithm for optimal APP symbol decoding of linear block codes," *IEEE Trans. on Inform. Theory*, vol. IT–44, no. 7, pp. 3124–3129, Nov. 1998.

[29] P. O. Vontobel, "Algebraic coding for iterative decoding," Ph.D. dissertation, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 2003.

[30] ——, "A factor-graph approach to the context-tree weighting method," in *Proc. IEEE Data Compression Conference*, Snowbird, UT, USA, 2004, p. 571.

[31] A. Vardy and F. R. Kschischang, "Proof of a conjecture of McEliece regarding the expansion index of the minimal trellis," *IEEE Trans. on Inform. Theory*, vol. IT–42, no. 6, pp. 2027–2034, Nov. 1996.