Some Remarks on Factor Graphs

Hans-Andrea Loeliger

Dept. of Information Technology and Electrical Engineering, Signal Proc. Lab. (ISI),
Swiss Federal Institute of Technology (ETH), CH-8092 Zürich, Switzerland.
E-mail: loeliger@isi.ee.ethz.ch

Abstract:
The paper is a collection of remarks, some rather plain, on various issues with factor graphs. In particular, it is pointed out that powerful signal processing techniques such as gradient methods, Kalman filtering, and the particle filter can be used and combined in factor graphs.

Keywords: factor graphs, turbo signal processing, gradient methods.

1 INTRODUCTION

Based on prior work by Wiberg et al. [1] [2], factor graphs were introduced in [3] and [4]. The main point of these papers was to show that many known algorithms in coding, artificial intelligence, and signal processing may be viewed as instances of the summary-product algorithm that operates by message passing in the factor graph.

Factor graphs can of course also be used to develop new algorithms for particular applications, but the literature on such applications is still quite limited; examples include [5] and [6].

For the past two years, we have been applying factor graphs to a wide variety of practical problems in areas ranging from communications over biomedical signal processing to fire detection devices. (Most of this is as yet unpublished, but see [7] and [8].) It has become apparent to us that the factor graph notation is indeed extremely helpful for the development of practical algorithms. A key issue in most such applications is the coexistence of discrete and continuous variables; another is the harmonic cooperation of a variety of different signal processing techniques. Some remarks on these topics are given in the present paper.

The paper is structured as follows. In Section 2, we briefly review Forney-style factor graphs (called *normal graphs* in [9]), which have become our preferred notation. Section 3 is a collection of remarks on some unrelated little issues that may be helpful to non-experts; none of this is new. In Section 4, we address some general issues with continuous variables. In Section 5, we outline the use of gradient methods for local message computations in factor graphs. Some conclusions are offered in Section 6.

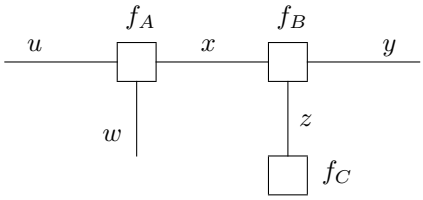

Figure 1: A Forney-style factor graph (FFG).

This paper is *not* an introduction to factor graphs; for such an introduction, see, e.g., [4] [9] [10] [11].

2 FORNEY-STYLE FACTOR GRAPHS

A *Forney-style factor graph* (FFG) is a diagram as in Figure 1 that represents the *factorization* of a function of several variables. E.g., assume that some function $f(u, w, x, y, z)$ can be factored as

$$f(u, w, x, y, z) = f_A(u, w, x) f_B(x, y, z) f_C(z). \quad (1)$$

This factorization is expressed by the FFG shown in Figure 1. In general, an FFG consists of nodes, edges, and "half edges" (which are connected only to one node), and there are the following rules:

- There is a node for every factor.
- There is an edge (or half edge) for every variable.
- The node representing some factor g is connected with the edge (or half edge) representing some variable x if and only if x is an argument of g.

Implicit in these rules is the assumption that no variable appears in more than two factors. This restriction is easily circumvented, however. For example, consider the factorization

$$f(x) = f_A(x) f_B(x) f_C(x). \quad (2)$$

We expand this into

$$f(x) = f_A(x) f_B(x') f_C(x'') \delta(x - x') \delta(x - x''), \quad (3)$$

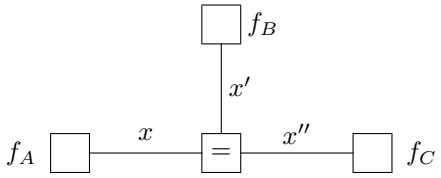

Figure 2: Cloning of variables.

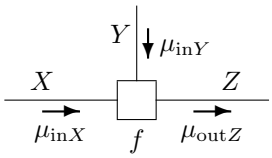

Figure 3: Message passing.

which is represented by the FFG in Figure 2. (The "δ" in (3) may be either a Kronecker delta or a Dirac delta, depending on the context.) The free use of auxiliary variables (such as x' and x'' in Figure 2) is typical for Forney-style factor graphs.

Other than in Figures 1 and 2, we will usually denote the variables by capital letters and specific values ("realizations") of the variables with lowercase letters.

For FFGs, the basic sum-product rule for the computation of messages has a particularly simple form. For example, in Figure 3, the message out of node f along the edge Z is

$$\mu_{\text{out}Z}(z) = \sum_x \sum_y f(x,y,z)\, \mu_{\text{in}X}(x)\, \mu_{\text{in}Y}(y). \quad (4)$$

We also recall that the sum in (4) may be replaced by other summary operators such as integration or maximization, cf. [4]. In any case, a message is a "summary" of the graph "behind" it.

3 SOME SIMPLE REMARKS

3.1 Combining Information

Consider a situation as in Figure 4, where two codes share the coded symbols X_k, $k = 1, 2, 3, \ldots$ In such cases, the correct handling of extrinsic information and intrinsic information is usually considered an issue that requires attention. Not so with factor graphs: the correct handling of extrinsic information and intrinsic information is automatic.

3.2 Mappers and Such

Consider a situation as in Figure 5, where two binary symbols, X_A and X_B, are mapped to a 4-AM symbol Z. Let $f : \mathbb{Z}_2 \times \mathbb{Z}_2 \to \{-3, -1, +1, +3\}$ be this mapping and assume that x_A is mapped to the more significant bit of z. In an FFG, the mapper becomes a factor node with local function

$$\delta_f(x_A, x_B, z) \triangleq \begin{cases} 1, & \text{if } f(x_A, x_B) = z \\ 0, & \text{else} \end{cases} \quad (5)$$

The computation of all messages in and out of the node (cf. Figure 6) is immediate from the sum-product rule (4). For example, we have

$$\mu_{\text{in}X_A}(x_A) = \sum_{x_B, z} \delta_f(x_A, x_B, z)\, \mu_{\text{out}X_B}(x_B)\, \mu_{\text{in}Z}(z), \quad (6)$$

which expands to

$$\begin{aligned} \mu_{\text{in}X_A}(0) = {} & \mu_{\text{out}X_B}(1) \cdot \mu_{\text{in}Z}(+3) \\ & + \mu_{\text{out}X_B}(0) \cdot \mu_{\text{in}Z}(+1) \quad (7) \\ \mu_{\text{in}X_A}(1) = {} & \mu_{\text{out}X_B}(0) \cdot \mu_{\text{in}Z}(-1) \\ & + \mu_{\text{out}X_B}(1) \cdot \mu_{\text{in}Z}(-3). \quad (8) \end{aligned}$$

3.3 Hybrid Equality Constraint

Consider an equality constraint between a variable X that takes values in some finite set $\mathcal{X}$ and a

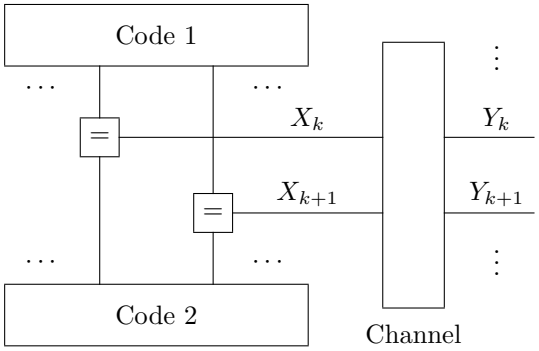

Figure 4: Shared code symbol.

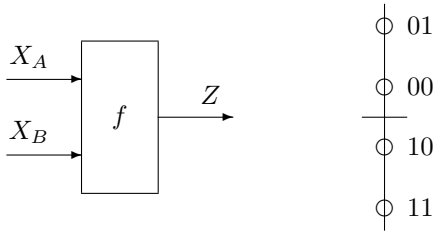

Figure 5: Bits-to-symbol Mapper.

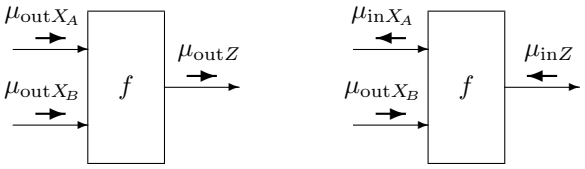

Figure 6: Messages through the mapper.

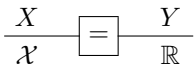

Figure 7: Hybrid equality node.

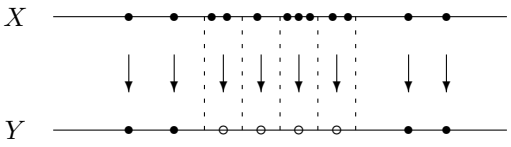

Figure 8: Quantizer.

real variable Y, cf. Figure 7. Such an equality constraint translates into a factor $\delta(x-y)$, which should be read as a Kronecker delta in x and a Dirac delta in y. According to the sum-product rule (4), the message out of the X-edge is

$$\mu_{\mathrm{out}X}(x) = \int_y \delta(x-y)\,\mu_{\mathrm{in}Y}(y) \tag{9}$$

$$= \mu_{\mathrm{in}Y}(x), \tag{10}$$

which amounts to sampling the incoming density $\mu_{\mathrm{in}Y}$ at the elements of $\mathcal{X}$. The message out of the Y-edge is

$$\mu_{\mathrm{out}Y}(y) = \sum_{x\in\mathcal{X}} \delta(y-x)\,\mu_{\mathrm{in}X}(x), \tag{11}$$

a sum of weighted Dirac deltas.

3.4 Quantizers

Let X be a variable that takes on values in some finite set $\mathcal{X}$. Consider a quantizer $q : X \to Y : x \mapsto q(y)$. The set $\mathcal{Y}$ of possible values of Y may be a finite subset of $\mathbb{R}$, or it may consist of subsets (intervals) of $\mathbb{R}$. Such a quantizer may be present in the original system or it may have been introduced to make some message computations more tractable (cf. Section 3.5). For the latter purpose, a quantizer as in Figure 8 may be attractive. The messages through such a quantizer node are easily computed to be

$$\mu_{\mathrm{out}Y}(y) = \sum_{x:q(x)=y} \mu_{\mathrm{in}X}(x) \tag{12}$$

and

$$\mu_{\mathrm{out}X}(x) = \mu_{\mathrm{in}Y}(q(x)). \tag{13}$$

3.5 Real-Sum Nodes

Consider the problem of computing messages through a node that represents the constraint

$$\sum_{\ell=0}^{m} X_\ell = 0 \tag{14}$$

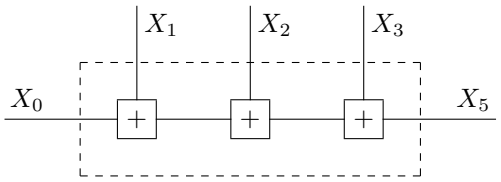

Figure 9: Sum constraint of several variables.

where, for $\ell > 0$, the variables X_ℓ take values in finite subsets $\mathcal{X}_\ell$ of $\mathbb{R}$, and X_0 takes arbitrary values in $\mathbb{R}$. The literal application of the sum-product rule (4) yields sums with about $\prod_{\ell=1}^{m} |\mathcal{X}_\ell|$ terms, which is infeasible for large alphabets and/or large m. Note that such a node may be decomposed as in Figure 9, but this decomposition does not, by itself, solve the complexity problem.

In practical applications, however, the computations can usually be reduced to a manageable level. One way to achieve this is to insert a quantizer (e.g., as in Figure 8) between the "small" sum constraint nodes in Figure 9. By properly adjusting such quantizers to the noise level of the application, the performance loss can usually be kept negligible.

In summary, sum constraints among finite-alphabet real variables can usually be handled computationally. One should not forget, though, that sum constraints among many variables dilute information: messages through such nodes tend to be uninformative.

4 REMARKS ON CONTINUOUS VARIABLES

For continous variables, literal application of the sum-product or max-product update rules often leads to intractable integrals. Dealing with continuous variables thus involves the choice of suitable message types and of the corresponding (exact or approximate) update rules. So far, the following message types have proved useful:

Constant messages. The message is a "hard-decision" estimate of the variable. This message type appears, e.g., if a decision-feedback equalizer is represented as a message passing algorithm. Another example is $\hat{\theta}$ in Section 5 below.

Quantized messages are an obvious choice (cf. Section 3.5). However, quantization is usually infeasible in higher dimensions.

Mean and variance of (exact or assumed) **Gaussian Messages.** This is the realm of Kalman filtering. Kalman filtering as message passing in a factor graph was briefly treated in [4] and

worked out in more detail in [10], see also [11] [12] [13].

The derivative of the message at a single point is the data type used for gradient methods, see Section 5.

List of samples. A probability distribution can be represented by a list of samples from the distribution. This data type is the basis of the *particle filter* [14]; its use for message passing algorithms in general graphs seems to be largely unexplored, but promising.

Note that all these message types are consistent with the axiom that a message is a summary of everything "behind" the transmitting node.

With these message types, it is possible to integrate most good known signal processing techniques as local message computations in a factor graph. In the next section, we outline such a translation for gradient methods.

5 ON GRADIENT METHODS

The use of gradient methods for local message computations in factor graphs is illustrated in Figure 10, which represents the global function $f(\theta) \triangleq f_A(\theta) f_B(\theta)$. The variable Θ is assumed to take values in $\mathbb{R}$ or in $\mathbb{R}^n$. Suppose we wish to find

$$\hat{\Theta} \triangleq \operatorname*{argmax}_{\theta} f(\theta) \tag{15}$$

by solving

$$\frac{d}{d\theta}\big(\log f(\theta)\big) = 0. \tag{16}$$

Note that

$$\frac{d}{d\theta}\big(\log f(\theta)\big) = \frac{f'(\theta)}{f(\theta)} \tag{17}$$

$$= \frac{f_A(\theta) f_B'(\theta) + f_B(\theta) f_A'(\theta)}{f_A(\theta) f_B(\theta)} \tag{18}$$

$$= \frac{d}{d\theta}\big(\log f_B(\theta)\big) + \frac{d}{d\theta}\big(\log f_A(\theta)\big). \tag{19}$$

Figure 10 may be a part of some bigger FFG. In this case, the nodes f_A and f_B may be summaries of the graph "behind" them. The functions f_A and f_B may be infeasible to represent, or to compute, in their entirety, but it may be easy to evaluate $\frac{d}{d\theta}\big(\log f_A(\theta)\big)$ (and likewise for f_B) at any given point θ.

A general gradient method to find a solution $\hat{\theta}$ of (16) operates as follows.

1. The equality constraint node in Figure 10 broadcasts some initial estimate $\hat{\theta}$. The node f_A replies by sending

$$\frac{d}{d\theta}\big(\log f_A(\theta)\big)\Big|_{\theta=\hat{\theta}}$$

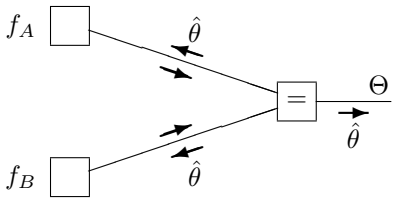

Figure 10: On gradient methods.

and the node f_B replies accordingly.

2. A new estimate $\hat{\theta}$ is computed as

$$\hat{\theta}_{\text{new}} = \hat{\theta}_{\text{old}} + s \cdot \frac{d}{d\theta}\big(\log f(\theta)\big)\Big|_{\theta=\hat{\theta}_{\text{old}}} \tag{20}$$

where $s \in \mathbb{R}$ is a positive step-size parameter.

3. The procedure is iterated as one pleases.

As always with message passing algorithms, there is much freedom in the scheduling of the individual operations.

It can be shown, for example, that the standard LMS algorithm may be obtained in this way from a suitable factor graph. If second derivatives are also available, Newton-type methods can be used.

6 CONCLUSIONS

- "Turbo signal processing"—iterative message passing in a graphical model—allows to combine and to extend many of the best known algorithms for detection and estimation problems, including gradient methods, Kalman filtering, and particle filters. Our accumulating experience confirms that the graphical framework helps to see such algorithmic options in practical problems.
- As factor graphs develop into a general tool for signal processing, there appear to be similar developments (with similar graphical models) in statistics, artificial intelligence, and neural networks, cf. [15] [13].
- Dealing with *continuous* variables involves many design choices, and this is a vast field of research. Nevertheless, many good practical solutions are apparent already.
- *Forney-style* factor graphs provide an especially elegant notation.

REFERENCES

[1] N. Wiberg, H.-A. Loeliger and R. Kötter, "Codes and iterative decoding on general graphs," *Europ. Trans. Telecomm.,* vol. 6, pp. 513–525, Sept/Oct. 1995.

[2] N. Wiberg, *Codes and Decoding on General Graphs.* Linköping Studies in Science and Technology, Ph.D. Thesis No. 440, Univ. Linköping, Sweden, 1996.

[3] B. J. Frey, F. R. Kschischang, H.-A. Loeliger, and N. Wiberg, "Factor graphs and algorithms," *Proc. 35th Allerton Conf. on Communications, Control, and Computing,* (Allerton House, Monticello, Illinois), Sept. 29 – Oct. 1, 1997, pp. 666–680.

[4] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Information Theory,* vol. 47, pp. 498–519, Feb. 2001.

[5] A. P. Worthen and W. E. Stark, "Unified design of iterative receivers using factor graphs," *IEEE Trans. Information Theory,* vol. 47, Feb. 2001, pp. 843–849.

[6] J. Boutros and G. Caire, "Iterative multiuser decoding: unified framework and asymptotic analysis," *Proc. 2001 IEEE Int. Symp. on Information Theory,* Washington DC, June 24–29, 2001, p. 317.

[7] J. Dauwels and H.-A. Loeliger, "Joint decoding and carrier synchronization using factor graphs," *Proc. 2003 IEEE Int. Symp. Information Theory,* Yokohama, Japan, June 29 – July 4, 2003.

[8] B. Vigoda, J. Dauwels, N. Gershenfeld, and H.-A. Loeliger, "Low-complexity LFSR synchronization by forward-only message passing," submitted to *IEEE Trans. Information Theory.*

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

[10] H.-A. Loeliger, "Least squares and Kalman filtering on Forney graphs," in *Codes, Graphs, and Systems,* R. E. Blahut and R. Koetter, eds., Kluwer, 2002, pp. 113–135.

[11] P. O. Vontobel and H.-A. Loeliger, "On factor graphs and electrical networks," in *Mathematical Systems Theory in Biology, Communication, Computation, and Finance,* J. Rosenthal and D. S. Gilliam, eds., IMA Volumes in Math. & Appl., Springer Verlag, to appear.

[12] P. O. Vontobel, *Kalman Filters, Factor Graphs, and Electrical Networks.* Internal report INT/200202, ISI-ITET, ETH Zurich, April 2002.

[13] M. I. Jordan, *Graphical Models,* draft of book.

[14] A. Doucet, J. F. G. de Freitas, and N. J. Gordon, eds., *Sequential Monte Carlo Methods in Practice.* New York: Springer-Verlag, 2001.

[15] M. I. Jordan and T.J. Sejnowski, eds., *Graphical Models: Foundations of Neural Computation.* MIT Press, 2001.