Dynamic Message-Passing Decoding on Non-bipartite Graphs

Hannah A. Hansen

Spring 2015

This master project, directed by Pål Ellingsen, Constanza Riera, and Matthew G. Parker, concerns itself with the implementation and potential benefits of dynamic message-passing on simple undirected graphs using $\mathbb{F}_4$-additive codes. We begin in this proposal by presenting some coding theory background and classical binary message-passing decoding on factor graphs, upon which this thesis will be brought together. Thereafter, we present the topic of inquiry in conjunction with the research questions of the thesis.

1 Introduction

Every day, whenever a phone call is made or the internet is accessed, someone somewhere has initiated the receiving process of data transmission over a communication channel. The use of communication channels to transmit information is embedded in our everyday lives. Walkie-talkies, hard drives, ROVs for deep-sea exploration, and satellites are a few examples of systems that depend on communication channels for core functionality. These channels may have a wide variety of physical manifestations, but in this proposal we will rely only upon an abstract definition in order to make our way towards the vital aspects of the project. A *communication channel* is any medium that allows for the transmission of data from one point to another, either through space or time.

The reliability and efficiency of these transmissions are fundamental concerns of digital communications, as channels frequently corrupt information transmitted, in which case they are called *noisy channels.* Consider the scenario where Paul and Amanda are caught on either side of the River Thames. Paul, the transmitter, wishes to communicate to Amanda, the receiver, that he is trapped and requires her assistance in breaking free before several guards return to transport him to the Tower of London. His only choice is to shout across the river and hope Amanda can comprehend the message and come to his aid. However, the guards manage to return before she becomes aware of what is the matter, as the noise of the river disturbs Paul's message and forces him to spend time repeating it.

One of the central problems of communication is *how to handle the presence of noise.* In the late 1940s, Shannon wrote a seminal treatise[2], in which one of his main insights was that noise imposed by a channel can be overcome with the use of channel coding. The main design behind channel coding, or forward error correction, is to add sufficient redundancy to a transmission, such that, in the event of transmission error, the receiver may recover the original message. Coding theory is the study of error-correcting codes, which are used to ensure reliable communication in the presence of noise. In this proposal we shall be dealing with two types of error-correcting codes, *binary linear codes* and $\mathbb{F}_4$-*additive codes.*

Linear codes

Linear codes are error-correcting codes where any linear combination of codewords yields a valid codeword of the code. They allow for efficient encoding and decoding algorithms, and are used frequently in channel coding.

Definition 1.1 (Linear code) *A linear code, $\mathcal{C}$, of length n and rank k over a field $\mathbb{F}_q$ is a linear subspace of the vector space $\mathbb{F}_q^n$.*

A binary linear code, $\mathcal{C}$, of size n and rank k, is a linear code over $\mathbb{F}_2 = \{0, 1\}$. Any linear code may be expressed by means of both generator matrices and parity-check matrices. A $(k \times n)$ generator matrix, G, for a code $\mathcal{C}$, is a basis for the vector space of codewords and may consist of any set of k linearly independent codewords. G also defines an encoding function such that, for any message $\mathbf{m}$ of length k, we have by the multiplication, $\mathbf{m}G$, a codeword of length n. We also have a means for determining whether or not a sequence of length n is a codeword of $\mathcal{C}$, namely the $(n - k \times n)$ parity-check matrix, H, defined as follows. H is any set of $(n - k)$ linearly independent row vectors of the dual code of $\mathcal{C}$, such that, for all $c \in \mathcal{C}$, we have that $cH^\top = 0$.

Example 1.1 *For the* $[7, 4, 3]_2$ Hamming code*, we have the following generator matrix, G, and parity-check matrix, H.*

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \qquad H = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

The Sum-Product Algorithm

The sum-product algorithm (SPA) is a message-passing algorithm for performing inference on graphical models, such as *Trellis graphs* and *Markov models.* When performed on what is known as a factor graph, the algorithm serves as a tool in error-detection and error-correction at the receiving end of a transmission. To better understand how it functions we must first familiarize the notion of a factor graph and show its connections to linear codes.

Factor graphs. A factor graph is a graphical representation of the factorization of a global function. Consider the function $g(X)$, where $X = \{x_0, x_1, x_2, x_3, x_4\}$, such that

$$g(X) = f_0(x_0)f_1(x_1)f_2(x_0, x_1, x_2)f_3(x_2, x_3)f_4(x_2, x_4).$$

Given a factorization of $g(X)$, we can now construct a factor graph, G. When doing so, we let every variable of $x_i \in X$ constitute a variable node, and every factorization f_i constitute a factor node. Moreover, for every $f_i(X_i)$ and every $x_j \in X_i$ we have the edge (f_i, x_j) in G.

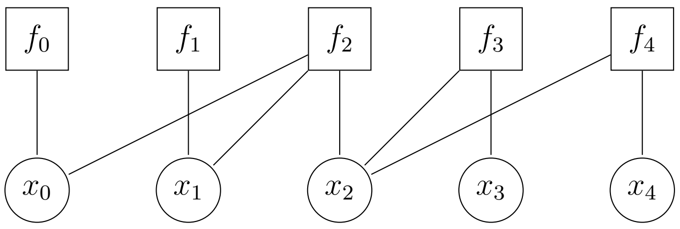

Figure 1: Factor graph G of $g(X)$.

In the context of binary linear codes, it is also possible to construct a factor graph from the parity-check matrix H. In this scenario, every row of the matrix represents a factor node and every column represents a variable node, such that, for all factor nodes f_i and variable nodes x_j, there is an edge between them if and only if the entry a_{ij} of the parity-check matrix is a 1.

Example 1.2 *Using the parity-check matrix of Example 1.1. we obtain the following factor graph.*

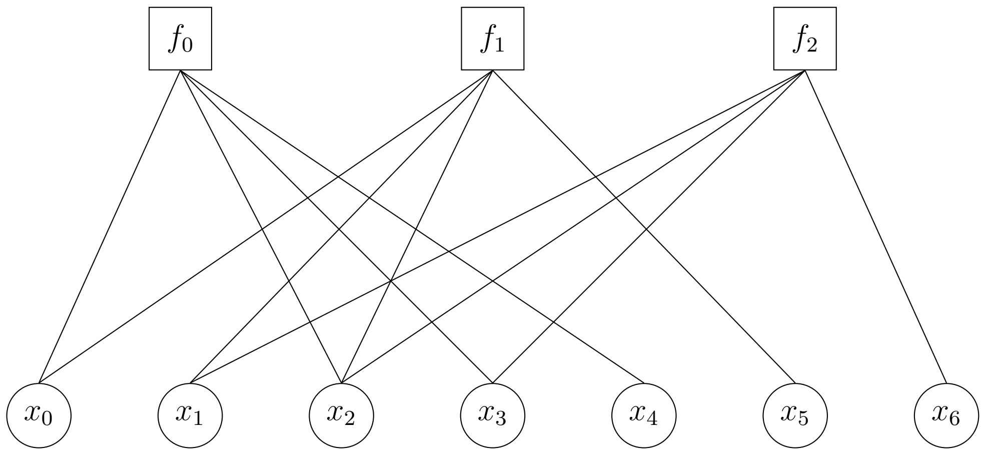

Figure 2: Factor graph of function in Example 1.1.

The SPA computes the marginals of various functions comprising the factor graph upon which it is executed. These marginals are then used to make a hard decisions about the state of each transmitted bit given the knowledge of the received codeword.

Once a transmission, r, is received from the channel, each variable node x_i is given an initial belief about its state, based on the received bit, r_i, and the channel model. The initial belief is a two-state vector $x_i = (a, b)$. Here $a = P(c_i = 0 \mid r_i)$, the probability of transmitting $c_i = 0$ given that the observed bit is r_i , and similarly for $b = P(c_i = 1 \mid r_i)$. The message-passing begins once the initial beliefs have been provided, but before we examine its procedure, we will familiarize the message calculation of each type of node.

Factor nodes. Factor nodes pass messages that concern their belief of the state of the recipient. The content of each such message, $f_i \to x_j$, is based on the information f_i has recieved from all of its other neighbors, except for x_j. The computations done at each factor node may be demonstrated using a truth table. On the left hand side we see the domain of the factor and all of the potential configurations of input cascading below. To the right, followed by the messages sent by x_l and x_n, we see the definition of the function, where 1 signifies valid input.

x_l	x_j	x_n	f_i	$x_l \to f_i$	$x_n \to f_i$	
1	1	1	0	a	c	0
1	1	0	1	a	d	ad
1	0	1	1	a	c	ac
1	0	0	0	a	d	0
0	1	1	1	b	c	bc
0	1	0	0	b	d	0
0	0	1	0	b	c	0
0	0	0	1	b	d	bd

We compute the message $f_i \to x_j$ by first calculating the product of each row of the columns representing f_i and the messages sent to it. Thereafter, we sum over the different values of x_j as demonstrated with the colors blue and red signifying 0 and 1, respectively.

$$f_j \to x_j = \begin{pmatrix} ac & + & bd \\ ad & + & bc \end{pmatrix}$$

Variable nodes. Variable nodes pass beliefs about themselves to their neighbors. For x_j to send a message to f_i, x_j must take all the 2-state vectors it has received from neighbors other than f_i and multiply them together. This belief then becomes equivalent of x_j telling f_i what everyone else believes about him.

x_j	$f_k \to x_j$	$f_m \to x_j$	$x_j \to f_i$
0	a	c	ac
1	b	d	bd

Algorithmic procedure.

It is possible to use the SPA to compute single marginals, but we will only be describing the scenario where each marginal is computed, as we are interested in the marginal of each bit of the received codeword. If the factor graph in question is cycle-free, i.e. is a tree, then the algorithm is exact and the marginal values always converge. Otherwise, the algorithm is not guarenteed to be exact. When cycles are present one runs the risk of getting stuck in an infinite loop, if the values do not converge inside the cylce. Furthermore, cycles may amplify false information by reverberating assumptions. We will explain these two scenarios separately, beginning with trees.

Trees. Choose an arbitrary node as the root node - either factor or variable. Begin message- passing at leaf nodes and propagate up towards the root. After the root has received messages from each child, begin message-passing from root through the rest of the tree towards the leaves. When the leaves have received messages, then one can calculate the marginals of each variable by multiplying the incoming and the outgoing messages of a single edge incident on each variable node.

Cycles. Here we exploit an iterative approach with no natural termination, and the general procedure is as follows. Assume an identity message has been sent on every edge in each direction. Thereafter, for each iteration, messages are passed in each direction on every edge until the values have converged, or a decided upon stop criteria has been met.

Given that messages from v to u depends on messages from all other neighbors of v, how are the messages propagated? We introduce the notion of pending messages. A node v has a message pending for u if v has received a message from any other neighbor after the latest transmission to u. In other words, receiving any message at v will cause a pending message along all other edges incident on v. If v has a pending message to u and has received all other messages, then v sends a message to u.

2 Dynamic Message-Passing on Simple Undirected Graphs

The area of inquiry for this master's thesis regards *message-passing decoding* over the *quaternary symmetric channel* as described in [1]. We will in this thesis implement a simulation tool for decoding $\mathbb{F}_4$-additive codes using sum-product message-passing on simple non-bipartite graphs, as opposed to working with bipartite factor graphs. The initial goal is to establish the feasibility of such an application. In the process we wish to quantify its efficiency. Thereafter, we wish to compare the simulation tool to a generic $\mathbb{F}_4$ decoding system and address the following claim made in [1].

> "Simple graph decoding is a special subset of 4-state decoding so, as a result, one has a more efficient decoder for simple graph decoding, as it only has to deal with the special case."

An $\mathbb{F}_4$-additive code, C, is different from an $\mathbb{F}_4$-linear code in that it is a code over $\mathbb{F}_4 = \{0, 1, \omega, \omega^2\}$, such that the only legal linear combinations are the ones of the form $v + u = c$, where $u, v, c \in C$. We will be concerned $\mathbb{F}_4$-additive codes that are self-dual with respect to the Hermitian inner product, as they may be represented with matrices of the form, $G = H = \omega I + A$. with these matrices we may construct simple nonbipartite graphs, upon which we intend to perform message-passing decoding. From this project, we expect to discover and map properties of $\mathbb{F}_4$-additive messag-passing decoding on nonbipartite graphs.

3 Research Methodology

The object of this research project is to verify feasibility and quantify the efficiency of the $\mathbb{F}_4$-additive message-passing decoding as described in [1]. In order to acheive this we will implement a simulation tool that will model all the real world aspects of a communication system that relies upon the quaternary symmetric channel. By excluding disturbing factors of the real world we isolate our method - allowing for an accurate and reliable study of its properties.

The feasibility of our method will established by confirming the correctness of an implemention of the message-passing decoding simulation tool itself. Thereafter, in order to quantify efficiency via the method of comparison, we intend to implement two variations of the same application. The application shall use message-passing in order to compute the maximum independent set of a graph. This can be done with generic decoding over the quaternary symmetric channel, in addition to dynamic message-passing decoding. Simulations will then be run in order to compare the two methods.

4 Thesis Outline

Introduction. This section will contain a basic introduction to channel coding and message-passing decoding. More specifically, it will explain message-passing decoding in light of binary and quaternary decoding.

Dynamic Messag-Passing Decoding. Here we will explain the method of dynamic message-passing decoding and provide a detailed explanation of the research problem at hand.

Project Outline. Before elaborating on the implementation we give a brief introduction to the outline of the project.

IMPLEMENTATION

Prototype. In this section we will introduce the implementation of the prototype as well as the trials associated with the process.

Simulation Tool. Here we will elaborate in detail on the architecture of the simulation tool as well as its functionality. Additionally, we will elaborate on the implmentation of the various applications implemented for comparison purposes.

Testing. An elaboration of the testing phase with respect to method and results will be provided in this section.

RESULTS

Simulations. Here we will explain the process of simulation Thereafter, we will present the results of the project along with suggestions for further research.

Conclusions. Lastly, we will provide the conclusions of the project, which are yet to be determined.

5 Project Plan

August to December

Implement Prototype. Initially, we intend to implement a rough prototype of the messge-passing decoder in order to gauge how the implementation of the actual simulator should be done.

Implement Simulator. Thereafter, we shall implement the simulation tool together with the two variatons of the max-independent set application.

Test phase.

January to Febuary

Simulations. In this phase, we shall run simulations of the decoding process with both application variants. Results will be gather in this phase as well.

March to June

Thesis Work. Lastly, the thesis will be written in the time frame from March to June.

6 Conclusions and Outlook

The process of formulating a project proposal has been a great source of structure for the project and yielded a clear direction for the project to continue in. Regardless of results, it will be a great joy to acquire the results of the planned simulations. The outlook seems to indicate an exciting process of learning and discovery.

References

[1] Matthew G. Parker - *Dynamic Message-Passing Decosing on Simple Graphs for the Quarternary Symmetric Channel*

[2] C. Shannon - *A Mathematical Theory of Communication, 1948, Bell System Technical Journal 27 (3): 379–423.*

[3] T. Richardson and R. Urbanke - *Modern Coding Theory*

[4] Siddhartha Biswas - *Introduction to Coding Theory: Basic Codes and Shannon's Theorem*

[5] Ian Stewart - *Galois Theory*

[6] John B. Fraleigh - *A First Course in Abstract Algebra*