Implementation in Python of Progressive Edge Growth algorithm to generate and visualize Tanner Graphs and Multi-Edge Type LDPC with Parity Check Matrix

by

Jona Bako

Bachelor Thesis in Computer Science

Submission: February 26, 2024

Supervisor: Thesis Supervisor

Constructor University | School of Computer Science and Engineering

Statutory Declaration

Family Name, Given/First Name	Bako, Jona
Matriculation number	30004124
Kind of thesis submitted	Bachelor Thesis

English: Declaration of Authorship

I hereby declare that the thesis submitted was created and written solely by myself without any external support. Any sources, direct or indirect, are marked as such. I am aware of the fact that the contents of the thesis in digital form may be revised with regard to usage of unauthorized aid as well as whether the whole or parts of it may be identified as plagiarism. I do agree my work to be entered into a database for it to be compared with existing sources, where it will remain in order to enable further comparisons with future theses. This does not grant any rights of reproduction and usage, however.

This document was neither presented to any other examination board nor has it been published.

German: Erklärung der Autorenschaft (Urheberschaft)

Ich erkläre hiermit, dass die vorliegende Arbeit ohne fremde Hilfe ausschließlich von mir erstellt und geschrieben worden ist. Jedwede verwendeten Quellen, direkter oder indirekter Art, sind als solche kenntlich gemacht worden. Mir ist die Tatsache bewusst, dass der Inhalt der Thesis in digitaler Form geprüft werden kann im Hinblick darauf, ob es sich ganz oder in Teilen um ein Plagiat handelt. Ich bin damit einverstanden, dass meine Arbeit in einer Datenbank eingegeben werden kann, um mit bereits bestehenden Quellen verglichen zu werden und dort auch verbleibt, um mit zukünftigen Arbeiten verglichen werden zu können. Dies berechtigt jedoch nicht zur Verwendung oder Vervielfältigung.

Diese Arbeit wurde noch keiner anderen Prüfungsbehörde vorgelegt noch wurde sie bisher veröffentlicht.

. . . .
Date, Signature

Abstract

Thesis Abstract (summarize purpose and content of thesis)

Contents

1 Introduction 1

2 Statement and Motivation of Research 4

3 Description of the Investigation 4

4 Evaluation of the Investigation 4

5 Conclusions 4

1 Introduction

Low-density parity-check (LDPC) codes represent a significant advancement in the realm of error-correcting codes, characterized by their sparse parity-check matrices and impressive decoding capabilities. Defined by an $m \times n$ matrix, where $n > M$ and $M = N - K$ (N: Total number of bits in a codeword; K: Number of information bits; M: Number of parity bits or check nodes.), LDPC codes are predominantly studied in their binary form, offering a compelling balance between performance and complexity in modern communication systems.

The crux of LDPC code construction lies in the creation of a sparse parity-check matrix, where the number of '1' entries is notably smaller than '0' entries. This sparsity is a defining feature, contributing to efficient decoding processes. The row-weight, denoted as k, signifies the number of '1's in a row, while the column-weight, denoted as j, indicates the number of '1's in a column. When both row and column weights remain constant, the LDPC code is considered regular; otherwise, it is irregular.

Constructing LDPC codes involves a meticulous process of determining the connections between rows and columns of the parity-check matrix or between check and symbol nodes within the corresponding Tanner graph. This process, crucial for achieving desired LDPC code parameters such as rate, girth, and length, aims at optimizing both decoding performance and hardware implementation. The fundamental goal is to design LDPC codes that offer robust error correction capabilities while ensuring ease of hardware integration.

In the pursuit of optimal LDPC code construction, the challenge arises in balancing the trade-offs between decoding performance and hardware complexity. Regular codes, with their structured row-column connections, often present easier hardware implementation; however, they may lack the flexibility needed for varied code designs. On the other hand, irregular codes, while offering greater design flexibility, can result in increased hardware complexity due to their unstructured interconnections.

Various construction methods exist, ranging from random (unstructured connections) to structured (predefined connections), each with its own set of advantages and limitations. Random constructions, despite their flexibility, introduce complexities in decoder interconnections, while structured constructions may limit the range of achievable rates, lengths, and girths. Thus, the quest continues for methods capable of generating a diverse range of LDPC codes that balance performance metrics with implementation constraints.

One prominent algorithm in LDPC code construction is the Progressive Edge Growth (PEG) algorithm. This non-algebraic method offers a simple yet effective approach to constructing LDPC codes of varying lengths and rates. The PEG algorithm, as described by Gabofetswe Alafang Malema in his Ph.D. thesis paper: "Low-Density Parity-Check Codes: Construction and Implementation", builds a Tanner graph by iteratively adding edges, ensuring minimal impact on the graph's girth. Notably, codes generated through the PEG algorithm have demonstrated superior performance, particularly at shorter code lengths.

v_l is a variable node l and f_l is check node l.
E_{v_l} is a set of edges connected to variable node v_l
d_{v_l} is the edge degree of variable node l
$\aleph_{v_l}^{g}$ is a set of check nodes reached by a subgraph spreading from variable node v_l with depth g.

for $l = 0$ to $N-1$ do {
  for $t = 0$ to d_{v_l}-1 do {
    if t =0 {
    $E_{v_l}^{0} \leftarrow edge\ (f_i, v_l)$, where $E_{v_l}^{0}$ is the first edge incident to variable node v_l, and f_i is one check node such that it has the lowest check-node degree under the current graph setting $E_{v_0} \cup E_{v_1} ... \cup E_{v_{l-1}}$
    }
    else { expand a subgraph from symbol node v_l up to depth g under the current graph setting such that the cardinality of $\aleph_{v_l}^{g}$ stops increasing but is less than M, or $\overline{\aleph}_{v_l}^{g} \neq \emptyset$ but $\overline{\aleph}_{v_j}^{g+1} = \emptyset$, then $E_{v_j}^{t} \leftarrow$ edge (f_i, v_l), where $E_{v_l}^{t}$ is the k^{th} edge incident to v_l, and f_i is one check node picked from the set $\overline{\aleph}_{v_l}^{g}$ having the lowest check node degree.
  } } }

Figure 1: Sample PEG Algorithm by Gabofetswe Alafang Malema

Further advancements in the PEG algorithm have been proposed, aiming to enhance the performance of the obtained codes. Hua Xiao and Amir H. Banihashemi introduced improvements to the standard PEG algorithm, focusing on selecting check nodes with higher connectivity to improve the resulting LDPC codes' performance. Additionally, Lin Chen and Da-Zheng Feng presented a fast implementation of the PEG algorithm, utilizing a red-black (RB) tree array to manage the check nodes efficiently.

The improved PEG algorithm by Hua Xiao and Amir H. Banihashemi introduces modifications to the original algorithm, particularly in selecting check nodes based on their connectivity to improve the resulting LDPC codes' performance. This modification, effective for constructing irregular codes, aims to enhance the connectivity of the Tanner graph, thereby improving the minimum distance and reducing trapping sets (short cycles within the Tanner graph), crucial for reducing errors in the error-floor region.

PEG algorithm [3],[5]
for $j = 0$ to $n-1$ do {
 for $k = 0$ to $d_{s_j} - 1$ do {
  if $k = 0${
   $E_{s_j}^{0} \leftarrow$ edge (c_i, s_j), where $E_{s_j}^{0}$ is the first edge incident to symbol node s_j, and c_i is one check node such that it has the lowest check-node degree under the current graph setting $E_{s_0} \bigcup E_{s_1} \bigcup \cdots \bigcup E_{s_{j-1}}$}
  else {
   expand a subgraph from symbol node s_j up to depth l under the current graph setting such that the cardinality of $N_{s_j}^{l}$ stops increasing but is less than m, or $\bar{N}_{s_j}^{l} \neq \phi$ but $\bar{N}_{s_j}^{l+1} = \phi$, then $E_{s_j}^{k} \leftarrow$ edge (c_i, s_j),

Figure 2: Improved Construction of PEG Algorithm by Hua Xiao and Amir H. Banihashemi

On the other hand, the fast implementation of the PEG algorithm by Lin Chen and Da-Zheng Feng offers a novel approach to constructing LDPC codes efficiently. By utilizing an RB-tree array to manage check nodes, this implementation significantly reduces computational complexity while maintaining the quality of generated codes. Through dynamic adjustments of the RB tree structure, the algorithm achieves efficient construction of LDPC codes with large girths, essential for improving error correction capabilities.

Require: $d(s_i) \leq d(s_j)\ \forall i < j$ and $f(c_i) = d(c_i)\ \forall i$
  for $j = 1$ to n **do**
    for $k = 1$ to $d(s_j)$ **do**
      if $k = 1$ **then**
        if $d(s_j) = 2$ **then**
          $E_{s_j}^1 \leftarrow (c_i, s_j)$, where $E_{s_j}^1$ is the first edge incident to s_j and c_i is a check node such that it has the *lowest check-node degree* under the current graph setting $E_{s_1} \cup E_{s_2} \cup \cdots \cup E_{s_{j-1}}$.
        else
          $E_{s_j}^1 \leftarrow (c_i, s_j)$, where $E_{s_j}^1$ is the first edge incident to s_j and c_i is a check node such that it has the *highest free check-node degree*.
        end if
      else
        Expand a subgraph from symbol node s_j up to depth l under the current graph setting, such that $\mathcal{N}_{s_j}^l = \mathcal{N}_{s_j}^{l+1}$, or $\overline{\mathcal{N}}_{s_j}^l \neq \emptyset$ but $\overline{\mathcal{N}}_{s_j}^{l+1} = \emptyset$.
        $E_{s_j}^k \leftarrow (c_i, s_j)$, where $E_{s_j}^k$ is the kth edge incident to s_j and c_i is a check node picked from the set $\overline{\mathcal{N}}_{s_j}^l$ having the *highest free check-node degree*.
      end if
      $f(c_i) = f(c_i) - 1$
    end for
  end for

Figure 3: Fast Implementation of PEG Algorithm by Lin Chen and Da-Zheng Feng

In this thesis, we aim to thoroughly explore and implement these advancements in the PEG algorithm using Python. The focus will be on the practical aspects of constructing the PEG algorithm developed by Hua Xiao and Amir H. Banihashemi. This includes creating Tanner graphs and multi-edge LDPC codes step by step, from the initial stages to the final output. Additionally, visualization methods will be developed to display these generated graphs based on user input. Through this detailed approach, a comprehensive understanding of how the algorithm works, its performance characteristics and its real-world applications in LDPC codes will be achieved.

2 Statement and Motivation of Research

This part should make clear which question, exactly, you are pursuing, and why your project is relevant/interesting. This is the place to explain the background and to review the existing literature. Where does your project extend the state of the art? What weaknesses in known approaches do you hope to overcome? If you have carried out preliminary experiments, describe them here.

(target size: 5-10 pages)

3 Description of the Investigation

This is the technical core of the thesis. Here you lay out your how you answered your research question, you specify your design of experiments or simulations, point out difficulties that you encountered, etc.

(target size: 5-10 pages)

4 Evaluation of the Investigation

This section discusses criteria that are used to evaluate the research results. Make sure your results can be used to published research results, i.e., to the already known state-of-the-art.

(target size: 5-10 pages)

5 Conclusions

Summarize the main aspects and results of the research project. Provide an answer to the research questions stated earlier.

(target size: 1/2 page)