Undergraduate Thesis

Hi-C interaction matrix correction using ICE in Rust

Felix Karg

Examiner: Prof. Dr. Backofen
Advisers: Dr. Mehmet Tekman, Joachim Wolff

Albert-Ludwigs-University Freiburg

Faculty of Engineering

Department of Computer Science

Chair for Bioinformatics

July 10th, 2019

Writing Period

10. 04. 2019 – 10. 07. 2019

Examiner

Prof. Dr. Backofen

Advisers

Dr. Mehmet Tekman, Joachim Wolff

Bachelor Thesis

Hi-C interaction matrix correction using ICE in Rust

Felix Karg

Gutachter: Prof. Dr. Backofen
Betreuer: Dr. Mehmet Tekman, Joachim Wolff

Albert-Ludwigs-Universität Freiburg

Technische Fakultät

Institut für Informatik

Lehrstuhl für Bioinformatik

10. Juli 2019

Bearbeitungszeit

10. 04. 2016 – 10. 07. 2016

Gutachter

Prof. Dr. Backofen

Betreuer

Dr. Mehmet Tekman, Joachim Wolff

Declaration

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare, that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Place, Date	Signature

Abstract

foo bar

Zusammenfassung

German version is only needed for an undergraduate thesis.

(TODO: Schreiben!)

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

List of Algorithms

1 Introduction

Hi-C is a well-known method for getting 3D-interaction information about genomes [1]. This information, however usually is quite biased (inherently based on the method), and can be corrected through an iterative method [2].

1.1 Algorithm

(TODO: describe the algorithm) **(EXTEND: THIS:)** Our fundamental assumption is that every location in our Matrix has in total as many interactions (with other locations) as every other location. Taking this in mind, the algorithm itself is pretty straightforward.

1.2 Operation

(DRAFT: Change Name!) smb can be run on any Unix-based operating system (tested using ubuntu-18.04 and macOS) with Python, Rust and common development packages installed (e.g. libopenssl-dev, python3-dev, build-essential, ...). For best performance, the size of the matrix should correlate with the number of available cores and the amount of available RAM **(EXTEND: Give rough factors!)**.

(TODO: Motivation as to why you'd want to do it at all and like this) **(TODO: remove original parts)**

This is a template for an undergraduate or master's thesis. The first sections are concerned with the template itself. If this is your first thesis, consider reading Section 1.5.

Of course, the structure of this thesis is only an example. Discuss with your adviser what structure fits best for your thesis.

1.3 Template Structure

- To compile the document either run the makefile or run your compiler on the file 'thesis_main.tex'. The included makefile requires latexmk which automatically runs bibtex and recompiles your thesis as often as needed. Also it automatically places all output files (aux, bbl, ...) in the folder 'out'. As the pdf also goes in there, the makefile copies the pdf file to the parent folder. There is also a makefile in the chapters folder, to ensure you can also compile from this directory.

- The file 'setup.tex' includes the packages and defines commands. For more details see Section 1.4.

- Each chapter goes into a separate document, the files can be found in the folder chapters.

- The bib folder contains the .bib files, I'd suggest to create multiple bib files for different topics. If you add some or rename the existing ones, don't forget to also change this in thesis_main.tex. You can then cite as usual

- The template is written in a way that eases the switch from scrbook to book class. So if you're not a fan of KOMA you can just replace the documentclass

in the main file. The only thing that needs to be changed in setup.tex is the caption styling, see the comments there.

1.4 setup.tex

Edit setup.tex according to your needs. The file contains two sections, one for package includes, and one for defining commands. At the end of the includes and commands there is a section that can safely be removed if you don't need algorithms or tikz. Also don't forget to adapt the pdf hypersetup!!
setup.tex defines:

- some new commands for remembering to do stuff:
  - `\todo{Do this!}`: **(TODO: Do this!)**
  - `\extend{Write more when new results are out!}`:
    (EXTEND: Write more when new results are out!)
  - `\draft{Hacky text!}`: **(DRAFT: Hacky text!)**
- some commands for referencing, 'in `\chapref{chap:introduction}`' produces 'in Chapter 1'
  - `\chapref{}`
  - `\secref{sec:XY}`
  - `\eqref{}`
  - `\figref{}`
  - `\tabref{}`

- the colors of the Uni's corporate design, accessible with
  `{\color{UniX} Colored Text}`
  - UniBlue
  - UniRed
  - UniGrey
- a command for naming matrices `\mat{G}`, $\mathbf{G}$, and naming vectors `\vec{a}`, $\mathbf{a}$. This overwrites the default behavior of having an arrow over vectors, sticking to the naming conventions normal font for scalars, bold-lowercase for vectors, and bold-uppercase for matrices.
- named equations:

```
\begin{align}
    d(a,b) &= d(b,a)\\ \eqname{symmetry}
\end{align}
```

$$d(a,b) = d(b,a) \tag{1}$$

symmetry

1.5 Advice

This section gives some advice how to write a thesis ranging from writing style to formatting. To be sure, ask your advisor about his/her preferences.

For a more complete list we recommend to read Donald Knuth's paper on mathematical writing. (At least the first paragraph). `http://jmlr.csail.mit.edu/reviewing-papers/knuth_mathematical_writing.pdf`

- If you use formulae pay close attention to be consistent throughout the thesis!

- Usually in a thesis you don't write 'In [24] the data is..'. You have more space than a paper has, so write 'AuthorXY et al. prepare the data... [24]'. Also pay attention to the placement: The citation is at the end of the sentence before the full stop with a no-break space. `... last word~\cite{XY}.`

- Pay attention to comma usage, there is a big difference between English and German. '...the fact that bla...' etc.

- Do not write 'don't ', 'can't' etc. Write 'do not', 'can not'.

- If an equation is at the end of a sentence, add a full stop. If it's not the end, add a comma: $a = b + c \quad (1)$,

- Avoid footnotes if possible.

- Use ``'' for citing, not "".

- It's important to look for spelling mistakes in your thesis. There are also tools like aspell that can help you find such mistakes. This is never an excuse not to properly read your thesis again, but it can help. You can find an introduction under `https://git.fachschaft.tf/fachschaft/aspell`.

- If have things like a graph or any other drawings consider using tikz, if you need function graphs or diagrams consider using pgfplots. This has the advantage

that the style will be more consistent (same font, formatting options etc.) than when you use some external program.

- Discuss with your advisor whether to use passive voice or not. In most computer science papers passive voice is avoided. It's harder to read, more likely to produce errors, and most of the times less precise. Of course there are situations where the passive voice fits but in scientific papers they are rare. Compare the sentence: 'We created the wheel to solve this.' to 'The wheel was created to solve this', you don't know who did it, making it harder to understand what is your contribution and what is not.

- In tables avoid vertical lines, keep them clean and neat. See 1 for an example. More details can be found in the 'Small Guide to Making Nice Tables' `https://www.inf.ethz.ch/personal/markusp/teaching/guides/guide-tables.pdf`

2 Related Work

Give a brief overview of the work relevant for your thesis.

(TODO: introduce original implementation in python and KU-decomposition in C++)

3 Background

Explain the math and notation.

(TODO: Cite/introduce/... the given papers, and introduce the required concepts)

Abbildung 1: Use tikz to draw nice graphs!

Algorithm 1 Stochastic Gradient Descent: Neural Network

Create a mini batch of m samples $\mathbf{x}_0 \dots \mathbf{x}_{m-1}$
foreach sample $\mathbf{x}$ **do**
  $\mathbf{a}^{\mathbf{x},0} \leftarrow \mathbf{x}$ ▷ Set input activation
  foreach Layer $l \in \{1 \dots L-1\}$ **do** ▷ Forward pass
    $\mathbf{z}^{\mathbf{x},l} \leftarrow \mathbf{W}^l \mathbf{a}^{\mathbf{x},l-1} + \mathbf{b}^l$
    $\mathbf{a}^{\mathbf{x},l} \leftarrow \varphi(\mathbf{z}^{\mathbf{x},l})$
  end for
  $\boldsymbol{\delta}^{\mathbf{x},L} \leftarrow \nabla_{\mathbf{a}} C_{\mathbf{x}} \odot \varphi'(\mathbf{z}^{\mathbf{x},L})$ ▷ Compute error
  foreach Layer $l \in L-1, L-2 \dots 2$ **do** ▷ Backpropagate error
    $\boldsymbol{\delta}^{\mathbf{x},l} \leftarrow ((\mathbf{W}^{l+1})^T \boldsymbol{\delta}^{\mathbf{x},l+1}) \odot \varphi'(\mathbf{z}^{\mathbf{x},l})$
  end for
end for
foreach $l \in L, L-1 \dots 2$ **do** ▷ ▷ Gradient descent
  $\mathbf{W}^l \leftarrow \mathbf{W}^l - \frac{\eta}{m} \sum_{\mathbf{x}} \boldsymbol{\delta}^{\mathbf{x},l} (\mathbf{a}^{\mathbf{x},l-1})^T$
  $\mathbf{b}^l \leftarrow \mathbf{b}^l - \frac{\eta}{m} \sum_{\mathbf{x}} \boldsymbol{\delta}^{\mathbf{x},l}$
end for

4 Approach

The approach usually starts with the problem definition and continues with what you have done. Try to give an intuition first and describe everything with words and then be more formal like 'Let g be ...'.

4.1 Choosing the right API to call Rust from Python

There are three main ways to execute Rust code from Python. In the following, common techniques are investigated.

One common way is rust-cpython. This library requires Rust 1.25 or higher (current versions are 1.33/34/35 for stable/beta/nightly respectively). Rust-cpython grants access to the python gil (global interpreter lock) with which Python code can be evaluated and Python objects modified. The resulting library (directly from compiled rust) can easily be imported into Python (but needs to be renamed). Native Rust code requires some wrapping first, as shown here:

```
#[macro_use] extern crate cpython;
use cpython::{PyResult, Python};
// add bindings to the generated python module
// N.B: names: "librust2py" must be the name of the `.so` or `.pyd` file
py_module_initializer!(librust2py, initlibrust2py, PyInit_librust2py, |py, m| {
    m.add(py, "__doc__", "This module is implemented in Rust.")?;
```

```
    m.add(py, "sum_as_string", py_fn!(py, sum_as_string_py(a: i64, b:i64)))
    Ok(())
});
// logic implemented as a normal rust function
fn sum_as_string(a:i64, b:i64) -> String {
    format!("{}", a + b).to_string()
}
// rust-cpython aware function. All of our python interface could be
// declared in a separate module.
// Note that the py_fn!() macro automatically converts the arguments from
// Python objects to Rust values; and the Rust return value back into a Py
fn sum_as_string_py(_: Python, a:i64, b:i64) -> PyResult<String> {
    let out = sum_as_string(a, b);
    Ok(out)
}
```

This kind of wrapping, though quite common and based on the Python C-API makes it hard to write idiomatic Code in Rust. Also, since Python is directly affected, the interactions with Python need to be considered while writing Rust-Code. In computer science one does usually not intentionally strive for complexity.

Another common approach is using the pyO3-library, which started off as a fork of rust-cpython, but has since seen quite drastic changes. For example, its using requires at least Rust version '1.30.0-nightly 2018-08-18'. This has been updated to '1.34.0-nightly 2019-02-06' with the most recently update. This is due to the usage of several unstable features, most of which have recently been able to be promoted to stable. Still missing is Specialisation though, which has at the time of writing still a long way to go. The library would also result in an easily importable (needs to be renamed first, still) cdylib (same as rust-cpython). The still intermingled way of writing the interface (certainly better but not by much compared to rust-cpython) as

well as the dependency on unstable nightly rust versions led to the decision of not using it either.

The third way, that is actually been promoted in the official Rust docs, is to generate a dylib and import that in python. No renaming necessary, but the communication between Rust and Python is a bit more low-level. The main wrapper is on the side of Python, transforming Arguments to Pointers and C-Representations, whilst the Rust part needs to conform to C-practices, which includes receiving a list by getting a pointer and the length of it. Other than that, the Rust code has some additional `\#[no_mangle]` and `\#[repr(C)]` (procedural) macros, which result in these parts actually accessible from e.g. Python or C. Since like this neither language depends on something only internal (or combinatorial), and both just depend upon the 'common, unchanging' C-interface, this seems to be the preferred way.

4.2 Problem Definition

Start with a very short motivation why this is important. Then, as stated above, describe the problem with words before getting formal.

4.3 First Part of the Approach

4.4 N-th Part of the Approach

(TODO: Introduce main approach, problems I came across and more)

5 Experiments

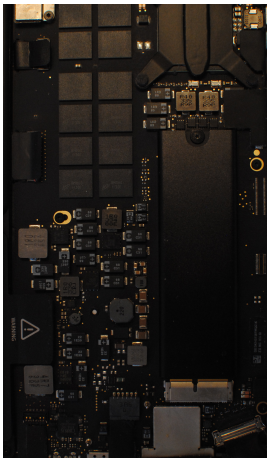
Type	Accuracy
A	82.47 ± 3.21
B	78.47 ± 2.43
C	84.30 ± 2.35
D	86.81 ± 3.01

Tabelle 1: Table caption. foo bar...

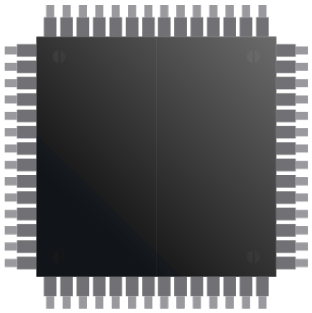
times (precomputed correction factors):

```
542.32user 447.26system 16:26.02elapsed 100\%CPU (0avgtext+0avgdata 116107220maxresident)k
14528inputs+8outputs (20major+159388812minor)pagefaults 0swaps
```

(TODO: The time-resource-measures done)



(a) Some cool graphic



(b) Some cool related graphic

Abbildung 2: Caption that appears under the fig Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

6 Conclusion

7 Acknowledgments

First and foremost, I would like to thank...

- advisers
- examiner
- person1 for the dataset
- person2 for the great suggestion
- proofreaders

ToDo Counters

To Dos: 9; 1, 2, 3, 4, 5, 6, 7, 8, 9

Parts to extend: 3; 1, 2, 3

Draft parts: 2; 1, 2

Literaturverzeichnis

[1] S. Wingett, P. Ewels, M. Furlan-Magaril, T. Nagano, S. Schoenfelder, P. Fraser, and S. Andrews, "Hicup: pipeline for mapping and processing hi-c data," *F1000Research*, vol. 4, 2015.

[2] M. Imakaev, G. Fudenberg, R. P. McCord, N. Naumova, A. Goloborodko, B. R. Lajoie, J. Dekker, and L. A. Mirny, "Iterative correction of hi-c data reveals hallmarks of chromosome organization," *Nature methods*, vol. 9, no. 10, p. 999, 2012.