Google Summer of Code Proposal

Canadian Centre for Computational Genomics

Author:Anđela Todorović

Mentor:Dr.Simon Gravel

Table of Contents

Project info

Project title: Large-scale modeling of genetic data

Project short title (30 characters): Genetic data modeling

URL of project idea page:
https://bitbucket.org/mugqic/gsoc_2020/src/master/#markdown-header-large-scale-modelling-of-genetic-data

Bio of Student

I'm a third-year student at the University of Niš, Faculty of Sciences and Mathematics, obtaining individual degrees in Computer Science and Mathematics. Ever since the start of my Bachelor's studies, I have determined my career goals and placed them toward the fields of statistical computation and data science, having a notable background as a competitor in Mathematics. After completing the first year of study, I have been offered a part-time job in a Swiss-based software company, where I have spent the subsequent year working as an industry researcher. During the course of that time, my study focus was on Deep learning applications on medical imagery and sensor data, prediction of functional ARMA processes and their utilisation in signal processing. One of the more extensive projects, that has been coordinated with European Respiratory Society, has involved lung tissue segmentation and cancer detection, where I have tested different clustering techniques for semantic segmentation, as far as Mask R-CNN model. Last summer, I interned as a researcher at Cubic Corporation, working on the implementation of a deep video classification network based on fine-tuned YOLOv3 architecture, for object detection. This project has caught the attention of LTS4 Signal processing lab at EPFL, where I will continue research on superresolution imaging of cancer tissue using generative adversarial networks in 2021. As a passionate contributor to the open-source community, my contribution has been conferred so far by Web Summit and Google in 2019.

Considering my previous work in the field of Biostatistics and a genuine interest towards processing of the genetic data, as far as graph theory and applications in big data processing, I do not doubt that I will be an extremely positive asset to your program. I certainly believe that my contributions to large repositories such as msprime and tskit would not only be highly beneficial to the future research in genomics, but also a great starting point for me to enter this domain in further education.

Contact information

Student name: Anđela Todorović

Student postal address: Vojvode Mišića 62/73, 18000 Niš, Serbia

Telephone(s): +381641386816/+381659752002

Email(s): andjelatodorovich@gmail.com

Other communications channels: Skype/Google+, etc. : Skype @andjellah

Student Affiliation

Institution: Faculty of Sciences and Mathematics, University of Nis

Program: Computer Science/Mathematics

Stage of completion: 3rd year of Bachelor of science studies

Contact to verify: Dean of the faculty, Dr. Perica Vasiljevic, dekan@pmf.ni.ac.rs

Schedule Conflicts: None

Mentors

Mentor names: Simon Gravel

Mentor emails: simon.gravel@mcgill.ca

Have you been in touch with the mentors? When and how?

After I have completed my sample test, I have contacted the Professor Gravel by email. We have scheduled a video call where we have discussed my background and interests, and where he has provided me with the suggestions of the potential project. The reconstruction of gene genealogies has immediately caught my attention, and so the Professor Gravel has followed up with additional papers on this subject, after which we had a broader discussion about the project workflow and the proposal.

Coding plan and methods

Describe in detail your plan for completing the work. What functions will be written, how and when will you do design, how will you verify the results of your coding? The subsection headings below are examples only. Each project is different, please make your application appropriate to the work you propose.

Tree sequences are a recently introduced, highly efficient way of storing a set of related DNA sequences by encoding their ancestral history as a set of correlated trees along the genome. This format is an output of various software libraries such as ***msprime***, whose primary goal is to simulate data efficiently and conveniently by generating coalescent trees for a sample under a range of evolutionary scenarios. The library itself is a reimplementation of Hudson's seminal ms program, and aims to eventually reproduce all of its functionality.

But the tree sequences can be used beyond simulation: they can also be used to encode real data and facilitate storage, sharing, and analysis the library tskit was developed to load, examine and manipulate tree sequences , The usage of tree sequences over Newick trees has allowed us to store and process very large scale simulation results efficiently.

Aside from genetic data, large amounts of historical data in the form of birth, death, and marriage records are being collected and digitized. Many datasets include thousands of records, and a few contain millions of linked genealogical records. This genealogical data is clearly related to the genetic data, since genomes are inherited through (biological) filiation. However, given genetic and genealogical data from the same individuals, identifying the precise path through which a given allele was inherited is computationally challenging. If we were able to do this, we would in effect reconstruct the genomes of the ancestors in the genealogy or, more precisely, the part of their genomes that were inherited by present-day samples. This is an important task in genetics that would open up many downstream applications, some of which are outlined below. From a computer science perspective, the goal is to embed a gene relatedness graph in the historical genealogy graph.

The objective of our proposal arises from the reconstruction of gene genealogies problem - from a given genotypes of sequence data from contemporary individuals and an extended pedigree of genealogical relationships among them, we have to decide and be very clear about what we should consider to be nodes information in genealogy tree according to kinds of dependencies among these nodes which should be considered to be edges of the tree.

In these evolving trees, traditional predictive models can not extract missing information, identify spurious interactions, evaluate evolving mechanisms, predict novel links and nodes attributes and so on, without taking into consideration structured knowledge correlations between

the objects. These node features can capture certain structural information, but they are not sufficient for capturing all qualitative aspects of a model.

This is one of the reasons why traditional models are inefficient and not able to deal with such a large amount of the required data.

Our goal is to build such a graph alignment method using the natural tree sequence encoding of the genetic data. This problem is far from resolved.A recent paper that tried to address such a graph alignment problem is [Ancestral Haplotype Reconstruction in Endogamous Populations using Identity-By-Descent, **link**]. but the authors didn't use the tree sequences and their approach did not scale to millions of individuals. The reason we believe this may now be possible is the introduction of the tree sequence format.

This project will focus on the development of novel algorithms for advanced reconstruction of the gene genealogy by performing inference on graphical models, specifically, implementing greedy and belief propagation algorithms on the succinct tree sequences.The implemented algorithms will in turn be integrated into the existing tskit and msprime libraries.

The succinct tree sequence is a recently introduced encoding for recombinant ancestry that takes advantage of the correlations between adjacent trees. Considering that tree sequence data structure has the potential to hugely reduce storage and processing costs, it is a natural (if not the only) contender to tackle this problem.

Phase 1. Greedy approach

Greedy approaches can be useful when there is a lot of information about a problem, but the computational cost of exploring all possibilities is prohibitive. The planned approach to this problem is to incrementally update the assignments in the graph alignment. At the starting point, we have a fixed number of individuals whose position in the gene genealogy and historical genealogy is known. Using these assignments as anchors, we will progressively add assignments (and, thus, anchors) via a heuristic. Each anchor represents an indication that a given genetic ancestor corresponds to a given genealogical ancestor. Whereas genealogical ancestors have two genealogical parents, genetic ancestral lineages are inherited from a single parent. The next assignment task is therefore to decide which of the two genealogical parents is also the genetic contributor. By iterating this process, we will obtain a graph alignment.

For this procedure to be useful, we need to make the right choices. However, we have a lot of information in our hands: from the genetic tree, we know the genetic and genealogical relatedness between the anchored node and all other anchors. We expect these to be strongly

correlated. We expect the correlation to strongly increase in the genetically contributing parent, and strongly decrease in the other parent. The greedy algorithm will therefore proceed by identifying, among all anchors, the anchor with the strongest differential in correlation among parents, and assign the high-correlation parent as the genetic ancestor.

Finally, the key to the computational efficiency of this approach is the rapid computation of correlations. Fortunately, this can be computed by a logarithmic number of updates for each greedy iteration.

In order to obtain a suitable model, and avoid unnecessary checking, we trim redundant sources: nodes whose individual multiset cardinality is not greater than the maximum cardinality of the multisets of its children. Once the redundant sources have been removed, we proceed by supposing that the source with the minimal distance to the cohort shares the gene sequence with the offspring, if no conflict occurs with this assignment. Thus, the same algorithm step shall be repeated until each of the individuals in the cohort (current last level of the tree) has been examined, after which the level will be deduced from the observation. This corresponds to the greedy choice of an optimal solution for the current level. At this point of work, we will introduce the pseudocode and mathematical description for the greedy algorithm into the paper, and implement the algorithm as an integral part of the tskit Python library .

Phase 2. Consideration of noisy data and performance tests

The second phase of the research will account for the significant presence of uncertainty and bias in the data, as induced in particular by adoption and mis-paternity. In terms of the graphical model, adoption results in linkages between individuals who likely don't share any biological similarities. At this phase, randomized testing will be performed to check the behavior of the greedy algorithm and identify feasible nodes at which the inconsistencies occur. This phase will reveal the correctness of the greedy approach as a placeholder, and depending on the obtained result, we will set the base conditions to the implementation of the belief propagation algorithm.

Phase 3. Loopy belief propagation algorithm

The third phase of the project that would be desirable to conduct, is to implement the modified loopy belief propagation algorithm (Pearl, 1988). The motivation behind this is to have a model which is based on mathematical reasoning, instead of heuristics, making it more precise, yet explainable.

The aim of the algorithm is to compute the marginal distribution at each node. Belief propagation performs by taking the form of a message-passing algorithm between nodes, expressed in terms

of an update to the outgoing message at iteration *i* from each node *t* to each neighbor *s* in terms of the previous iteration's incoming messages from *t*'s neighbors.

Concretely, when observing the gene genealogy, it would be extremely useful to define a transmitted message in terms of the number of steps between the observed individual at level l and each of the nodes at the level l+1, supposing we start the algorithm from the parent generation of offsprings in the last layer. As proven in [Understanding Belief Propagation and its Generalizations, link], eventually, the algorithm will converge at so called fixed points, which correspond to the stationary points of the Kullback-Leibler divergence problem. Once the algorithm converges, we will have one or more possible solutions to the problem.

Phase 4. Presenting advantages in the real-world applications

Considering that gene genealogy reconstruction has a variety applications, such as the investigation of genetic aberrations, which I have previously discussed with the Professor Gravel, starting from detecting point mutations up to larger rearrangements, it would be useful to perform a comparative analysis on the different approaches to this problem. Succinct tree sequences have shown the notable advance in terms of the complexity of the algorithm, and it is undeniable that our algorithms will find their part in the tskit and msprime projects.

Phase 5. Testing and documenting the code

This step will be performed throughout the length of the project to ensure code is working correctly. Two types of testing will be performed, including accuracy tests, and performance tests. Documenting the functions will be made using sphinx.

Accuracy tests include comparing the reconstructed genomes to their true values, while performance tests compare the algorithms efficiency to the current state of the art.

The final testing will be performed as well and the documentation of the code will either be merged with the existing msprime/tskit documentation, or published separately.

Timeline

Provide a detailed timeline of how you plan to spend your summer. Don't leave testing and documentation for last, as that's almost a guarantee of a failed project.

1. **May 4 - May 10**
    - First post-acceptance discussion with the mentor and the rest of the team in the C3G.
    - Forming even more detailed schedule plan and communicating about the possible challenges in the project.
    - Collecting literature about the project in order to get deeper domain knowledge, get familiar with the terminology and better understand common problems in genetics.
2. **May 11 - May 17**
    - Start reading about the gene genealogies from gained sources and some previous papers that have been provided by the mentor. Understanding and connecting both philosophical and practical views on the relationship between genetic and genealogical ancestry.
    - Detailed start of the research about the current state-of-the-art algorithms and their implementations and questioning scalability.
3. **May 18 - May 24**
    - Getting familiar with the graphical representation of genetic data and the common techniques used in big data processing.
    - Reading tskit and msprime documentation and samples. Installing them to my local environment and trying out provided examples.
    - Consultations with the mentor on choosing appropriate sample datasets, starting from the smaller ones, and increasing in size; discussing possible scaling issues and how to overcome them efficiently
4. **May 25 - May 31**
    - Performance testing of the state-of-the-art at the smaller datasets using existing methods in the tskit implementation.
    - Further research on succinct tree sequences from the available resources (code examples and pull requests on the msprime from the original creators, published papers in 2016 and 2018)

5. **June 1 - June 7**
    - Representing the provided data in the form of succinct tree sequences, and visualising the graph in order to detect separate marginal trees, loops and possible inconsistencies.
    - Repeating the same process on the large-scale dataset, without visualisation.
    - Setting up the coding environment. By this point, I will get introduced to tskit and msprime methods and their usage, so the project will be built upon the existing libraries and follow the same code organisation style.
6. **June 8 - June 14**
    - Start implementing the Greedy algorithm as described in the coding plan and methods section.
    - Implement the pseudocode and provide mathematical explanation.
    - Regularly commit the code and consult the mentor and other contributors for the code review.
7. **June 15 - June 21**
    - Continue with the Greedy algorithm implementation.
    - Fix potential bugs found in the previous week.
8. **June 22 - June 28**
    - Test the accuracy of the Greedy algorithm, and perform code optimisation. Reviewing code explainability and avoiding code duplications.
    - Test for scalability of the algorithm and explore different techniques for reducing the time complexity of the algorithm.
    - Briefly document the code.
9. **June 29 - July 5**
    - Start preparing for the first evaluation phase
    - Provide visible results that have been well documented and tested at this phase.
    - Write an additional paragraph about the Greedy algorithm on visibly noisy data.
10. **July 6 - July 12**
    - Review the first phase with the mentor and discuss plans for the upcoming phase
    - Start research on the loopy belief propagation from available resources and compare the algorithm to peeling-based algorithms.
11. **July 13 - July 19**
    - Fix the bugs which have been found during testing.
    - Dealing with noise or lackages in the data. Perform checking for conflicts and weak points of the Greedy approach at the large-scale dataset.
    - Suggest further improvements as the starting point of the belief propagation approach.

12. July 20 - July 26

- Try to implement the loopy belief propagation on the smaller dataset first. Notice stationary points and try to mathematically understand where the algorithm converges.
- Create a good pseudocode and start working on a supporting paper.
- Measure and compare the performance of the loopy BP. Review and try to document at this phase.

13. July 27 - August 2

- Start preparing for the second evaluation phase
- If applicable, provide results of the belief propagation approach that have been well documented and explained, along with the visualisations.
- Write a documentation that is richer in details.

14. August 3 - August 9

- Review the second phase with the mentor and discuss plans for the upcoming research.
- Start discussing the applications of the algorithm in genetic aberrations and try to find all relevant work in this field.
- Continue working on a supporting research paper.
- If needed, perform final tests of the algorithm.

15. August 10 - August 16

- By this point, sections on the Greedy approach and loopy backpropagation should have been added to the supporting paper, along with illustrations and pseudocodes.
- Finalizing the section about the real-life applications of the algorithms.

16. August 17 - August 23

- Discussion with the mentor about possible future work on the same topic.
- Summarizing and preparing for the final evaluation phase.
- Celebration :)

What is your contingency plan for things not going according to schedule?

I have left enough time in each of the time slots for the possible schedule changes. Still, if this happens despite my maximum dedication to the project, I will consult my mentor to publish the results on the smaller dataset or the set scaled to some extent, and leave possible extreme cases for the future work.

Management of coding project

How do you propose to ensure code is submitted / tested?

I will commit the code regularly to my GitHub profile: https://github.com/kobrica and the msprime branch. I have discussed submitting code to Dominic Nelsons branch of msprime https://github.com/DomNelson/msprime with my mentor.

All of the code will be public and open to code reviews from the organisation members, and all of the changes are to be made accordingly to the reviews and comments.

How often do you plan to commit? What changes in commit behavior would indicate a problem?

Since a large part of the project is focused on research, commits will be made once a week or more frequently, with the exception of the third phase of the project where additions will be made mostly to the supporting research paper. Commit behavior is unlikely to change, but more than 5 working days without commit would indicate an issue.

Test

Describe the qualification test that you have submitted to your project mentors. If feasible, include code, details, output, and examples of similar coding problems that you have solved.

The qualification tests can be found at my GitHub profile in the following repository.https://github.com/kobrica/ComputationalGenomicsGSoC

The f_st_allels notebook introduced the different F-statistics across populations in the 1000 Genomes project.The scikit-allel package has been used to compute the reduction in heterozygosity due to population structure, Fst , with the packaged functions first, but we have also computed it ourselves with the Weir-Cockerham estimator. The next task was to compare pairwise Fst across human populations, which I found similar to correlation matrices and PCA I have previously worked with. It was interesting to deduce useful observations about the population based on Fst distances. The task has also reviewed inbreeding coefficients Fis among different populations.

The Wright-Fisher notebook focused on a Wright-Fisher simulation model for studying the evolution of allele frequencies in a simple population. It explored the parameters necessary for defining populations, various ways one can choose to model evolution, and focused on the

statistics behind the model, which was the part that has drawn my attention immediately. I have used different elementary statistics and probability theoretical knowledge to determine the type of distribution expected from a certain problem definition (binomial,hypergeometric, Poisson), and calculating the expected value for allele frequencies and fixation.