SAHap

Zhaofeng Li

June 2020

1 Abstract

SAHap is a haplotype assembly program written in C++ that solves the Weighted Minimum Error Correction (wMEC) problem based on the simulated annealing algorithm, without directly optimizing the MEC value itself. It aligns fragments of chromosomes to recover the complete haplotype sequences. In this report, I will describe the underlying problem, summarize the work that has been done, and finally evaluate preliminary results. This report also includes material from my past two reports to provide a comprehensive description of the approach, and is also written partly as an introduction to the code itself.

2 Background

A haplotype refers to the set of alleles found on the same chromosome, inherited from the same parent. DNA sequencing machines produce paired fragments of sequences on a chromosome ("end reads"), in which the locus ("site") of the alleles are known but not the chromosome. End reads may overlap with each other, with the average number of times a site is covered defined as the "coverage." In the process of haplotype assembly, we reconstruct the complete haplotype sequences for an individual by assigning end reads to the correct chromosomes.

In our current experiments, we only deal with diploids with 2 haplotype sequences, like humans. Humans are diploids with 2 sets of chromosomes, each coming from a parent. The result of the haplotype assembly process for humans is therefore two sequences. By convention, the two haplotype sequences are often binary strings, as the possible alleles at each locus is known. The most common variant is assigned 0 (reference), with the alternative assigned 1. There may be extremely uncommon variants, which are not considered by such implementations.

3 Dataset

In our experiments, we used the synthetic dataset from the GenHap team.[1] The dataset contains generated diploid end read data for different numbers of

SNPs and coverage rates along with the ground truth, with the largest one covering 20,000 SNPs at 60x coverage (`20000SNPs_60x`). The input files are in the WhatsHap Input Format (WIF) initially conceived by the WhatsHap[2] team, with the output format being a simple text file with each line representing a haplotype sequence. The haplotypes are not ordered in the output file.

To generate smaller datasets with fine-grained control on the number of SNPs, we used a Python script to "crop out" regions of desired sizes from the largest file. Two of such derived datasets were used in preliminary evaluation. This script is included in the source repository as `reduce-dataset.py`.

4 The Minimum Error Correction Problem

The Minimum Error Correction Problem (MEC), proposed by Lippert et al., is NP-hard.[3] When different reads assigned to the same chromosome overlap, their letters on the same site may differ. The MEC value refers to the total number of letters different from the solution. In other words, the MEC value for a particular alignment is the minimum number of letter changes (corrections) one needs to make in order to achieve perfect alignment. Because the haplotype assembly process is often done on binary strings where individual digits may flip, the problem is also called Minimum Letter Flips (MLF).

In addition to the alleles, the DNA sequencer may also provide information on "the confidence level of the sequencer machine's reading."[4] Such level may be converted to a value between 0 to 1, and used as a weight when counting the minimum corrections. This variant is named the Weighted Error Correction Problem, or wMEC for short.

5 Simulated Annealing And the Objective Function

In simulated annealing, the state is perturbed by a random change (move) in each iteration, with each change accepted or rejected according to probabilistic function computed from the amount of loss (energy) the change decreases. The initial state of SAHap is created with each read assigned to a chromosome at random. In each iteration, SAHap randomly selects a read in the system and moves it to another chromosome, recomputing the MEC value in the process. The energy value is computed by dividing current MEC with the maximum possible value.

The system accepts all moves that cause a decrease in the overall energy, and has a higher chance of accepting moves that cause a smaller amount of increase in energy (bad moves). As time passes, the temperature decreases, and so does the chance of accepting bad moves. In a correct implementation of simulated annealing, the probability of accepting a bad across temperatures resembles the sigmoid function.

Instead of optimizing against the raw MEC value directly, we implemented a probability-based function, based on the coverage at a given position and the estimated sequencer error rate. In this approach, we model a certain letter in the sequence being a miscall as a Poisson event, with the probability of occurrence equal to the sequencer error rate. The objective function is the sum of the probabilities at each site where the disagreements are simply due to read errors. An alternative to this approach would be to compute the probabilities of each read being correct ("read-based objective"), since all letters always move together when we assign a read to a haplotype.

SAHap follows an object-oriented approach in implementing the algorithm. The state of the system is kept in a `Genome` object, which holds a number of `Haplotype` objects consistent with the organism's ploidy. State perturbation in the simulated annealing process is carried out by moving reads between `Haplotype`s. To facilitate testing, the user controls the runtime directly by specifying the desired number of iterations.

6 Visualization

Using JupyterLab with the Xeus Cling C++ kernel,[5] we have implemented a rudimentary way to ease the visualization of the simulated annealing process of SAHap. In the Jupyter environment, the user is able to interactively execute individual steps in the simulated annealing pipeline and create plots with third-party libraries like `matplotlib-cpp`.

7 Results

The current performance of SAHap leaves a lot to be desired. While SAHap appears to be able to determine a reasonable temperature schedule for all runs, when tested against a range of small number of SNPs, only about 60% of runs converge closely to the ground truth ("success" is defined as having $< 1\%$ error rate). The error rate of the rest of the runs appear to spread randomly between 10% and 50%.

When we attempt to unify the differences between runtimes by plotting the error rate against the overall progress of the run (between 0 and 1), there appears to be no discernable difference between the successful and unsuccessful runs up until 30% of the run, when the error rate starts to become nearly constant.

References

[1] A. Tangherloni, S. Spolaor, L. Rundo, M. S. Nobile, P. Cazzaniga, G. Mauri, P. Liò, I. Merelli, and D. Besozzi, "GenHap: a novel computational method based on genetic algorithms for haplotype assembly," *BMC Bioinformatics*, vol. 20, p. 172, Apr. 2019.

[2] M. Patterson, T. Marschall, N. Pisanti, L. van Iersel, L. Stougie, G. W. Klau, and A. Schönhuth, "WhatsHap: Weighted Haplotype Assembly for Future-Generation Sequencing Reads," *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology*, vol. 22, pp. 498–509, June 2015.

[3] R. Lippert, R. Schwartz, G. Lancia, and S. Istrail, "Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem," *Briefings in Bioinformatics*, vol. 3, pp. 23–31, Mar. 2002. Publisher: Oxford Academic.

[4] S.-H. Kang, I.-S. Jeong, H.-G. Cho, and H.-S. Lim, "HapAssembler: A web server for haplotype assembly from SNP fragments using genetic algorithm," *Biochemical and Biophysical Research Communications*, vol. 397, pp. 340–344, June 2010.

[5] Q. Community, "Xeus Cling," 2019. original-date: 2017-04-20T15:35:44Z.