SAHap: Attacking the wMEC Problem With Probability-based Objective Function
(a tiny bit closer)

Zhaofeng Li

March 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. In this report, we present continued work on SAHap, including corrections to the core annealing routine as well as a new probability-based objective function. From a set of test runs involving varying numbers of SNPs and run time limits, we hypothesize that the time required to satisfactorily perform haplotype assembly ($HE < 1\%$) grows exponentially with regards to the number of SNPs, although this claim cannot be verified with current data.

2 Background

A haplotype refers to the set of alleles found on the same chromosome, inherited from the same parent. DNA sequencing machines produce lossy end reads, paired fragments of sequences on a chromosome, with occasional errors in the results. End reads may overlap with each other, with the average number of times a site is covered defined as the "coverage." Through the process of haplotype assembly, we align the individual end reads to reveal the complete haplotypes for an individual.

3 Temperature Schedule Calibration

A new mechanism for determining the optimal temperature schedule for the simulated annealing process has been implemented, similar to the one used in SANA. Before the actual simulation is run, the program runs 10,000 iterations with the temperature fixed at various points to determine the two temperature endpoints for the simulation from p_{bad}. Specifically, the program first steps down in temperature until p_{bad} drops below .99, then slowly increases the temperature

to find the value for T_0. The ending temperature T_1 is determined similarly, with the p_{bad} threshold set at 10^{-10}.

In the meantime, we keep track of the time taken to perform the iterations to estimate the processor speed (i.e. the average number of iterations it can perform in one second), and use it to calculate the total number of iterations to perform based on the run time limit specified by the user. However, the estimation turned out to not be very accurate and would cause the simulation to run much longer than specified (about 2x in testing). This might be caused by processor throttling under sustained load. To mitigate this problem, we chose to divide the original estimated number of total iterations by 2.

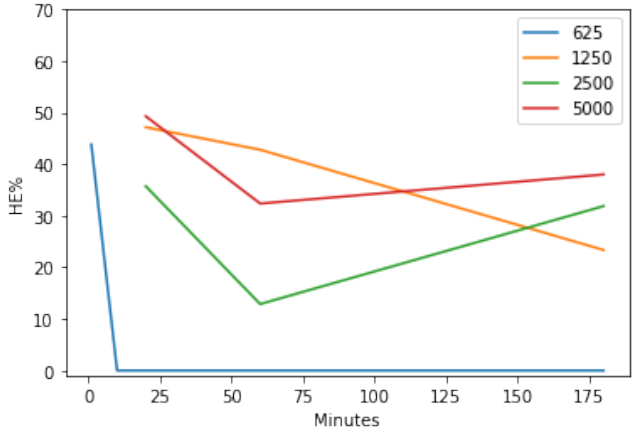
4 Probability-based Objective Function

During one of the meetings we looked at HapCHAT, a haplotype assembly program based on dynamic programming. Instead of directly optimizing the MEC value (number of disagreements), HapCHAT computes the probability that the given alignment is the correct solution, based on the estimated error rate of the sequencing machine.

In SAHap, we implemented a variant of this approach ("site-based objective"), 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 ("correct" for our purposes). 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.

5 Determining Required Run Time

To find out the relationship between the number of sites and the required run time, we designed a test matrix with varying number of SNPs and run times, with run times increasing exponentially at each step. The test dataset is derived from the synthetic 20,000 SNP files made by the GenHap project, with regions of the desired sizes "cropped out" from the original file. For each SNP count, the temperature endpoints are fixed (same input file), and the only variable is the allotted run time. The tests were done on the ICS openlab with the name `eat-cpu.sh`, much to the dismay of other users. With the inaccuracy in run time estimation (see Temperature Schedule Calibration), all of the runs took slightly longer than the specified time.



Each series in the graph correspond to an input file (fixed number of sites). Note that additional tests were added for the 625-SNP file as the program could perfectly reconstruct the ground truth with longer run times.

6 Conclusion

In summary, we made slight progress in roughly three main areas during the quarter. We improved the temperature schedule generation process, experimented with a new probability-based objective function, as well as performed test runs with various SNP counts and run times. All things considered, although some progress was made, it was not a particularly productive quarter. In hindsight, hard-to-debug errors and a lack of high-level awareness of the overall problem caused a decline in motivation.