Parallel Individual Haplotying Assembly : Xeon Phi vs. Nvidia K20x

Robert J. clucas

School of Electrical & Information Engineering, University of the Witwatersrand, Private Bag 3, 2050, Johannesburg, South Africa

Abstract:

Key words: Brach, Bound, GPU, Haplotyping, Simplex

1. INTRODUCTION

It is commonly accepted that all humans share ∼99% of the same DNA, however, small variations cause human beings to have different physical traits. Single nucleotide polymorphisms (SNPs), which are variations of a single DNA base from one individual to another, are believed to be able to address genetic differences. For diploid organisms, which have pairs of chromosomes, a *haplotype* is a sequence of SNPs in each copy of a pair of chromosomes. A *genotype* describes the conflated data of the haplotypes on a pair of chromosomes. Haplotypes are believed to contain more generic information than genotypes [1], however, obtaining haplotypes correctly is a difficult problem, which is broken into two subdomains: individual haplotype assembly and haplotype inference.

Haplotype inference uses the genotype of a set of individuals. The genotype data tells the status of each allele at a position, but does not distinguish which copy of the chromosome the allele came from. This negative aspects of this approach are that it cannot distinguish rare and novel SNPs [2], and there is no way of knowing if the inferred haplotype is completely correct.

Individual haplotype assembly uses fragments of sequences generated by sequencing technology to determine haplotypes. The fragments of a sequence come from the two copies of an individual's chromosome, the goal of the individual haplotyping problem is to correctly determine two haplotypes, where each haplotype corresponds to one of the two copies of the chromosome.

The haplotype assembly problem was proven to be NP-Hard [3]. The algorithms used to solve the problem are thus computationally complex and until recently, there was no practical exact algorithm to solve the problem using minimum error correction (MEC) [4], However, recently an exact solution was proposed by [5] which is capable of solving the MEC problem exactly, and can thus correctly infer all haplotypes from the fragment sequences. Due the NP-Hardness of the problem, the algorithm results in long run times - in the range of days for chromosomes with high errors rates. Using a parallel implementation of any of the proposed solutions could reduce the long run times, allowing useful haplotype information to be quickly inferred from the available datasets, having positive effects in fields such as drug discovery, prediction of diseases, and variations in gene expressions, to name a few.

Come back to introduction ...

Parallel programming makes use of devices which have many simple cores, but which can execute the same instructions on each of the cores at the same time. The effectiveness of parallel programming is dependant on the nature of the problem, as per Amdahl's law. The first attempts at parallel programming came from Graphics Processing Units (GPUs) which were used to render many pixels simultaneously. More recently, General-Purpose GPU (GPGPU) programming has become prominent with API's like CUDA and OpenCL, allowing access to GPUs from C and C++ programs.

2. BACKGROUND

2.1 Individual Haplotype Assembly Problem

This subsection will provide a brief overview of the individual haplotype assembly problem, and define the notation used through the rest of the paper. The input to the problem is a set of reads from a given genome sequence, where each read contains fragments from each of the two chromosomes which make up the genome sequence. These characters of a read consist of elements from a *ternary string*, where a ternary string has characters from the set {0, 1, -}. A value of 0 refers to the major allele at a site, a value of 1 to the minor allele, and a value of - to the lack of a read at the site. These reads are then combined to form a matrix, M, where each row of the matrix corresponds to a read.

Each column of the matrix is known as an SNP site. At each site, the data could be accurate, missing, or have error. The goal of the individual haplotype assembly problem is to determine a haplotype, H = {*h*,*h'*} from the matrix. The following terminology will be used to refer to properties of the matrix and the fragments.

For the input matrix M, the number of fragments is denoted by n, which is the number of rows in M. The number of SNP sites is doneted by m, which is the number of columns in M, while the j^{th} position of the i^{th} fragment is given by f_{ij}. Two fragments are said to conflict if the following conditions are true:

- $f_{ik} \neq f_{jk}$ **and** $f_{ik} \neq$ '-' **and** $f_{jk} \neq$ '-'

2.1.1 Minimum Fragment Removal

REFERENCES

[1] J. Stephens. "Haplotype Variation and Linkage Disequilibrium in 313 Human Genes." *Science*, vol. 293, no. 5529, pp. 489–493, Jul. 2001. URL `http://dx.doi.org/10.1126/science.1059431`.

[2] D. He, A. Choi, K. Pipatsrisawat, A. Darwiche, and E. Eskin. "Optimal algorithms for haplotype assembly from whole-genome sequence data." vol. 26, no. 12, pp. i183–i190, 2010.

[3] R. Lippert, R. Schwartz, G. Lancia, and S. Istrail. "Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem." vol. 3, no. 1, pp. 23–31, 2002.

[4] P. Bonizzoni, G. Della Vedova, R. Dondi, and J. Li. "The Haplotyping problem: An overview of computational models and solutions." *Journal of Computer Science and Technology*, vol. 18, no. 6, pp. 675–688, 2003. URL `http://dx.doi.org/10.1007/BF02945456`.

[5] Z.-Z. Chen, F. Deng, and L. Wang. "Exact algorithms for haplotype assembly from whole-genome sequence data." vol. 29, no. 16, pp. 1938–1945, 2013.