Parallelizing the Individual Haplotying Assembly Problem

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 (IHA) 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, and is referred to as a *gap*. 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 m, which is the number of rows in M. The number of SNP sites is denoted by n, which is the number of columns in M, while the j^{th} site of the i^{th} fragment is given by f_{ij}. Furthermore, two fragments are said to conflict if the following conditions are true:

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

Essentially this means that for two fragments i and j, if at an SNP site k, the reads do not have error and are not gaps, the reads have different values (fragment i has a value 0 at site k, while fragment j has a value 1 at site k, or vice versa).
Following the notion of a conflict, the *distance* between two fragments (or ternary strings) is denoted by d(f_i, f_j) is the total number of positions for which the two fragments f_i and f_j conflict. Furthermore, to understand some of the problem formulations from the fragment data, it is useful to define a *conflict graph* [6] G = { V, E }, where V corresponds to a fragment, and E corresponds to and edge between two fragments if they conflict. If the matrix M contains no errors, then none of the fragments from the same chromosome will conflict and G will be bipartate. Fragments from the same chromosome may conflict. However, if there are errors (as is usually the case) in M, G will not be bipartate. The individual haplotype assembly problem then requires the correction of G from a non-bipartate graph to a bipartate graph. There are numerous methods for doing this, and will be discussed in the subsections to follow.

2.1.1 Minimum Fragment Removal (MFR) The method of minimum fragment removal is to remove the smallest number of fragments from the inputs data such that the resultant graph G is bipartate. It is shown in [6] that this can be solved in polynomial time.

2.1.2 Minimum Edge Removal (MER) This method is not an exact error model [7], however, allows the problem to be reduced to a Max-Cut or weighted Max-Cut problem, which can then be solved to determine a bipartate graph. Informally, minimum edge removal requires determining the least number of edges from G such that removal of the edges results in G being bipartate.

2.1.3 Longest Haplotype Reconstruction (LHR) This requires finding a set of fragments which, when they are removed from M, result in G being bipartate and the length of the resultant haplotypes being maximized [7].

2.2 Minimum Error Correction (MEC)

This method involves correcting the minimum number of elements in the matrix M (sites for all fragments) which allow the graph G to be bipartate. Although being the most complex method, it is the most widely used method as it provides the highest accuracy rates. Furthermore, MEC is the only method which for which an exact solution to the individual haplotype assembly problem. For these reasons, a MEC solution was chosen to be parallelized.

2.3 Minimum Error Correction Implementations

Much research has been done on solving the MEC formulation of the IHA problem. The first exact algorithm for solving the problem was a branch and bound algorithm proposed by [8]. The algorithm creates a tree which covers the search space of all possible corrections. The tree is then traversed to find the best solution. They use an upper bound for when branching that allows branches to be abandoned when a better solution than the current best cannot result from further exploration of the branch, thus improving performance. However, this exact method has time complexity of $O(2^m)$, where m is the number of fragments in the input data, and hence cannot produce solutions for large problem sizes. They also provide a heuristic *genetic algorithm* (GA) to improve the computational time, which only requires run times up to three orders of magnitude faster than the branch and bound implementation. While the GA implementation gives very similar results to the branch and bound implementation, it is slightly worse.
A dynamic programming solution was proposed by [9] which addresses the run time problem for large input sizes. The algorithm has time complexity of $O(mk3^k + mlogm + mk)$, where k is the maximum number of SNP sites a fragment covers. In practice k is usually small, however, for inputs where k is large the algorithm becomes impractical.

REFERENCES

[1] J. Stephens. "Haplotype Variation and Linkage Disequilibrium in 313 Human Genes." *Science*, vol. 293, no. 5529, pp. 489–493, Jul. 2001. [Online]. Available at `http://dx.doi.org/10.1126/science.1059431` [Accessed 27 June 2015].

[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. [Online]. Available at `http://dx.doi.org/10.1007/BF02945456` [Accessed 27 June 2015].

[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.

[6] G. Lancia, V. Bafna, S. Istrail, R. Lippert, and R. Schwartz. "SNPs Problems, Complexity, and

Algorithms." In F. auf der Heide, editor, *Algorithms ESA 2001*, vol. 2161 of *Lecture Notes in Computer Science*, pp. 182–193. Springer Berlin Heidelberg, 2001. [Online] Available at `http://dx.doi.org/10.1007/3-540-44676-1_15` [Accessed 16 July 2015].

[7] R. Schwartz. "Theory and Algorithms for the Haplotype Assembly Problem." *Commun. Inf. Syst.*, vol. 10, no. 1, pp. 23–38, 2010. [Online] Available at `http://projecteuclid.org/euclid.cis/1268143371` [Accessed 16 July 2015].

[8] R.-S. Wang, L.-Y. Wu, Z.-P. Li, and X.-S. Zhang. "Haplotype reconstruction from SNP fragments by minimum error correction." vol. 21, no. 10, pp. 2456–2462, 2005.

[9] M. Xie, J. Wang, and J. Chen. "A Practical Exact Algorithm for the Individual Haplotyping Problem MEC." In *BioMedical Engineering and Informatics, 2008. BMEI 2008. International Conference on*, vol. 1, pp. 72–76. May 2008.