GenHap: A Novel Computational Method Based on Genetic Algorithms for Haplotype Assembly

Andrea Tangherloni[1,*], Simone Spolaor[1,2], Leonardo Rundo[1,3], Marco S. Nobile[1,2], Ivan Merelli[4], Paolo Cazzaniga[2,5], Daniela Besozzi[1], Giancarlo Mauri[1,2] and Pietro Liò[6]

[1]Department of Informatics, Systems and Communication, University of Milano-Bicocca, Viale Sarca 336, 20126 Milan, Italy
[2]SYSBIO.IT Centre of Systems Biology, Piazza della Scienza 2, 20126 Milan, Italy
[3]Institute of Molecular Bioimaging and Physiology, Italian National Research Council, Contrada Pietrapollastra-Pisciotto, 90015 Cefalù (PA), Italy
[4]Institute of Biomedical Technologies, Italian National Research Council, Via Fratelli Cervi 93, 20090 Segrate (MI), Italy
[5]Department of Human and Social Sciences, University of Bergamo, Piazzale Sant'Agostino 2, 24129 Bergamo, Italy
[6]Computer Laboratory, University of Cambridge, Cambridge CB3 0FD, United Kingdom
*andrea.tangherloni@disco.unimib.it

Introduction

Somatic human cells are diploids, that is, they contain 22 pairs of homologous chromosomes and a pair of sex chromosomes, one copy inherited from each parent. In order to fully characterize the genome of an individual, the reconstruction of the two distinct copies of each chromosome, called haplotypes, is mandatory. The process of inferring a full haplotype of a cell is known as haplotyping, which consists in assigning all heterozygous Single Nucleotide Polymorphisms (SNPs) to exactly one of the two chromosomes. SNPs are one of the most studied genetic variations, since they play a fundamental role in many medical applications, such as drug-design or disease susceptibility studies. Obviously, the complete set of SNPs of an individual (i.e., his/her haplotype) is generally more informative than analyzing single SNPs, especially in the study of complex disease susceptibility.

Nowadays, two classes of methods exist for haplotype phasing. The first class consist of statistical methods that try to infer haplotypes from genotypes sampled in a population. These data, combined with datasets describing the frequency by which SNPs are usually correlated in different populations, can be used to reconstruct the haplotype of an individual. The second class of methods analyzes sequencing read data, exploiting an idea similar to genome assembly. Their goal is to divide the entire set of reads into k partitions, corresponding to the k different haplotypes. In the case of diploid organisms, there are 2^n possible haplotypes for n heterozygous SNP positions.

The Minimum Error Correction (MEC) is one of the most used and promising approaches for haplotype assembly. This problem consists in computing the two haplotypes that partition the sequencing reads into two unambiguous sets with the least number of corrections to the SNP values [2]. Unfortunately, MEC was proven to be an NP-hard problem [1]. A weighted variant of MEC, named wMEC, was also proposed: in this version, the correction process takes into account the weight associated with each SNP value of a read [3]. The weights represent the confidence for the presence of a sequencing error and the read mapping to that particular genome position.

In this work, in order to tackle the computational complexity of the problem under examination, we propose GenHap, a novel computational method for haplotype assembly based on Genetic Algorithms (GAs). GenHap could efficiently solve large instances of the wMEC problem, yielding optimal solutions by means of a global search process.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.3246v1 | CC BY 4.0 Open Access | rec: 12 Sep 2017, publ: 12 Sep 2017

Methods

In a diploid organism, a read (also called fragment) is a vector $\mathbf{f} \in \{0, 1, -\}^n$ and a haplotype can be denoted as a vector $\mathbf{h} \in \{0, 1\}^n$. The value $-$ is used to indicate an uncovered element in a read, since the length of SNP fragments might be smaller than n. Given a set of n different SNPs and m fragments, we can define an $m \times n$ matrix $\mathbf{M}$, named fragment matrix.

Two distinct fragments $\mathbf{f}$ and $\mathbf{g}$ are in conflict if there is a position j, with $1 \leq j \leq n$, such that $f_j \neq g_j$ and $f_j \neq g_j \neq -$, otherwise they are in agreement. $\mathbf{M}$ is conflict-free if there are two different haplotypes $\mathbf{h}_1$ and $\mathbf{h}_2$ such that each row M_i (with $i = 1, \ldots, m$) is in agreement with either $\mathbf{h}_1$ or $\mathbf{h}_2$.

Finally, we can detect the case when two distinct fragments are in conflict by defining a distance $D(\cdot, \cdot)$ that measures the number of different values between two fragments $\mathbf{f} = (M_{i1}, \ldots, M_{in})$ and $\mathbf{g} = (M_{l1}, \ldots, M_{ln})$ of $\mathbf{M}$ (with $i, l \in \{1, \ldots, m\}$):

$$D(\mathbf{f}, \mathbf{g}) = \sum_{j=1}^{n} d(f_j, g_j), \tag{1}$$

where $d(f_j, g_j)$ is defined as:

$$d(x, y) = \begin{cases} 1, & \text{if } x \neq y,\ x \neq -, \text{ and } y \neq -, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

If $\mathbf{M}$ is conflict-free and there are no errors in all reads, $\mathbf{M}$ can be divided into two disjoint partitions $\mathbf{M}_1$ and $\mathbf{M}_2$ containing the fragments that are conflict-free, thus allowing to infer the two haplotypes, the first one from $\mathbf{M}_1$ and the second one from $\mathbf{M}_2$. Let $\mathbf{h}_1$ and $\mathbf{h}_2$ be two possible haplotypes, $\mathbf{M}_1$ and $\mathbf{M}_2$ the partitions of $\mathbf{M}$ representing the classification of all SNP fragments. We can infer $\mathbf{h}_1$ and $\mathbf{h}_2$ from $\mathbf{M}_1$ and $\mathbf{M}_2$, respectively, as follows:

$$h_{k_j} = \begin{cases} 1, & \text{if } N_1(\mathbf{M}_k)^j \geq N_0(\mathbf{M}_k)^j, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where $j = 1, \ldots, n$ and $k \in \{1, 2\}$. $N_0(\mathbf{M}_k)^j$ and $N_1(\mathbf{M}_k)^j$ denote the number of 0s and 1s in the j-th column, respectively. In such a way, $\mathbf{N}_0(\mathbf{M}_k)$ is the vector consisting of the number of 0s of each column j using the reads of the partition $\mathbf{M}_k$, while $\mathbf{N}_1(\mathbf{M}_k)$ is the vector consisting of the number of 1s of each column j represented by the partition $\mathbf{M}_k$. In order to solve the wMEC problem, $\mathbf{N}_1(\mathbf{M}_k)$ and $\mathbf{N}_0(\mathbf{M}_k)$ measure the weight associated with each SNP in each read. The distance $D(\cdot, \cdot)$ given in Equation 1 can be used also to evaluate the distance between a fragment and a haplotype. We define the following error function:

$$E(\mathbf{M}_1, \mathbf{M}_2, \mathbf{h}_1, \mathbf{h}_2) = \sum_{k=1}^{2} \sum_{\mathbf{f} \in \mathbf{M}_k} D(\mathbf{f}, \mathbf{h}_k). \tag{4}$$

By minimizing this error function, we can obtain the best partitioning of $\mathbf{M}$, inferring $\mathbf{h}_1$ and $\mathbf{h}_2$ with the least number of errors. Equation 4 is used as fitness function in GenHap.

GAs are a population-based optimization strategy that mimics Darwinian processes [4, 5, 6]. In GAs, a population P of randomly generated individuals undergoes a selection mechanism and is modified by genetic operators. In the most common formulation, each individual of P encodes a possible solution of the optimization problem as a fixed-length string of characters taken from a finite alphabet. Based on a quality measure (i.e., the fitness value), inspired by Darwin's "survival of the fittest" laws, each individual is involved in a selection process. The individuals characterized by good fitness values have a higher probability to be selected for the next generation; these individuals will generate possibly improved offspring thanks to the application of the crossover and mutation operators.

A general scheme that represents an implementation for GAs can be summarized as follows:
1) a population of individuals representing solutions of a given problem is randomly generated;
2) the fitness function is evaluated for each individual;
3) the individuals are selected according to their fitness value;
4) crossover and mutation operators are applied to recombine and vary the selected individuals;
5) if a chosen termination criterion is not satisfied, go back to step 2; otherwise, the individual characterized by the best fitness value is returned.

In the implementation of GenHap, we exploit a very simple and efficient structure for individuals, each one representing a partition of $\mathbf{M}$, encoded as a binary string. So doing, the length of each individual C_p, $p = 1, 2, \ldots, |P|$, of the population P is equal to m, and each of its elements is a binary value referring to a read. In order to obtain the two partitions $\mathbf{M}_1$ and $\mathbf{M}_2$, the individual C_p is evaluated as follows: if the i-th bit is equal to 0 then the read i belongs to $\mathbf{M}_1$, otherwise the read belongs to $\mathbf{M}_2$. Once the two partitions are computed, GenHap infers the haplotypes $\mathbf{h}_1$ and $\mathbf{h}_2$ by applying Equation 3. Finally, Equation 4 is used to calculate the number of errors made by exploiting the partition encoded by each individual of P. In our approach, we used a population composed of 100 individuals.

Among the different selection mechanisms employed by GAs (e.g., roulette wheel, ranking, tournament), GenHap exploits the tournament selection to create an intermediate population P', starting from P. In each tournament, T individuals are randomly selected from P and the individual having the best fitness value is added to P'. The size of the tournament T is related to the selection pressure. If T is large, the individuals characterized by worse fitness values have a lower probability to be selected, therefore decreasing the variability of P'; moreover, multiple copies of the same individual could be added to P'. In our test we set $T = \lfloor 0.2 \cdot |P| \rfloor$.

Afterwards, the genetic operators (i.e., crossover and mutation) are applied to the individuals belonging to P' to obtain the offspring for the next generation. The crossover operator allows for the recombination of two parent individuals (e.g., C_1 and C_2), generating the offspring that possibly has better characteristics with respect to the parents (in this work, crossover is applied with probability $c_r = 0.99$). In order to blend exactly half characters from C_1 and half from C_2, we use a single-point crossover resulting in a mixing ratio fixed to 0.5. The crossover procedure consists in selecting a crossover point $cp \in \{1, \ldots, m-1\}$, and exchanging 50% of the elements of C_1 with C_2, thus obtaining two new offspring. Finally, to increase the variability of the individuals, one or more elements of a selected offspring can be modified by applying the mutation operator (in this work, mutation is applied with probability $m_r = 0.01$). As a first mutation operator, we used a classic mutation in which a random number rnd is generated in $[0, 1)$ for each element e of the selected offspring, with $e = 1, \ldots, m$. If rnd is lower than m_r, the element e is mutated by flipping its value (i.e., from 0 to 1 or vice-versa). In addition to the prevoius mutation operator, in GenHap we implemented a bit-flipping mutation in which a random number of elements of the individuals are mutated according to m_r. In particular, this mutation is applied if the fitness value of the best individual does not improve for a given number of generations (5 in our implementation).

As the computational time required by GenHap increases with the number of reads and SNPs considered in the instance of the problem, to efficiently solve the wMEC problem we split the matrix $\mathbf{M}$ in sub-matrices consisting of γ reads. The $\lfloor m/\gamma \rfloor$ sub-matrices are then processed by means of a *divide-et-impera* approach that exploits a Master-Slave parallel programming paradigm.

Results

As a preliminary investigation of the performance of GenHap, we considered a matrix $\mathbf{M}$ of synthetic data consisting in 1088 reads and 549 SNPs. GenHap was then executed by varying the number of reads γ inserted in each sub-matrix and computing the minimum, maximum and mean errors (see Equation 4), as well as the required running time. The value of γ was varied in the range $[10, 90]$. Each test was repeated 30 times to achieve a statistical relevance. As shown in Table 1, the best results in terms of mimimum, mean and maximum number of errors are obtained with $\gamma = 30$, $\gamma = 50$ and $\gamma = 80$. As expected, the running time increases linearly with respect to the number of reads γ in which the matrix $\mathbf{M}$ is partitioned.

Conclusions

The reconstruction of haplotypes represents a hot topic in Computational Biology and Bioinformatics. In this paper we present GenHap, a novel computational method based on GAs to solve the haplotyping problem. In order to evaluate the effectiveness of our approach, we run GenHap on a common benchmark data set, consisting of synthetic genomic data. Our preliminary experimental results show that GenHap is able to achieve correct solutions in short running times. Moreover, this approach can be used to compute haplotypes in organisms with different ploidity.

Although several approaches have been proposed in literature to solve the haplotyping problem [7,

Table 1: Results achieved by GenHap on input matrix $\mathbf{M}$ with 1088 reads and 549 SNPs. Boldface values denote the best results in terms of error measures and computational time.

γ	Min error	Max Error	Mean Error	Running Time [s]
10	1	124	29.7	1.362
20	1	124	33.8	1.601
30	**1**	**1**	**1**	**1.813**
40	1	124	17.4	2.211
50	**1**	**1**	**1**	2.534
60	1	137	10.07	2.736
70	1	137	5.53	3.158
80	**1**	**1**	**1**	3.419
90	1	491	17.33	3.887

8, 9], the evolutionary technique introduced in this work has the advantage that it could be formulated and extended using a multi-objective fitness function. As a matter of fact, in addition to the number of flips to obtain an optimal partition, other objectives could be introduced in the fitness function, such as the methylation patterns of the different chromosomes or the gene proximity in maps achieved through Chromosome Conformation Capture (3C) experiments [10]. The final goal of leveraging multi-objective optimization (e.g., using NSGA-III [11]) is one of the reasons why we did not base GenHap on classic single-objective optimization methods, such as Simulated Annealing.

As a future extension of this work, we plan to compare GenHap against other methods typically employed in this field, such as WhatsHap [7] and HapCol [9]. In particular, we will apply GenHap on real data to the aim of thoroughly investigating its computational performance and effectiveness. For this goal, we will test GenHap using the data provided by the 1000 Genomes Project [12].

References

1. Lippert, R., Schwartz, R., Lancia, G., et al. Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. Briefings in Bioinformatics, 2002, 3(1):23–31.
2. Wang, R.S., Wu, L.Y., Li, Z.P. Haplotype reconstruction from SNP fragments by minimum error correction. Bioinformatics, 2005, 21(10):2456–2462.
3. Greenberg, H.J., Hart, W.E., Lancia, G. Opportunities for combinatorial optimization in computational biology. INFORMS Journal on Computing, 2004, 16(3):211–231.
4. Golberg, D.E. Genetic algorithms in search, optimization, and machine learning. Addison Wesley, 1989.
5. Baker, J.E. Adaptive selection methods for genetic algorithms. Proceedings of the 1st International Conference on Genetic Algorithms, 1985, 101–111.
6. Miller, B.L., Goldberg, D.E. Genetic algorithms, tournament selection, and the effects of noise. Complex Systems, 1995, 9:193–212.
7. Patterson, M., Marschall, T., Pisanti, N., et al. WhatsHap: weighted haplotype assembly for future-generation sequencing reads. Journal of Computational Biology, 2015, 22(6):498–509.
8. Bracciali, A., Aldinucci, M., Patterson, M., et al. PWHATSHAP: efficient haplotyping for future generation sequencing. BMC Bioinformatics, 2016, 17(11):342.
9. Pirola, Y., Zaccaria, S., Dondi, R., et al. HapCol: accurate and memory-efficient haplotype assembly from long reads. Bioinformatics, 2015, 32(11):1610–1617.
10. Merelli, I., Liò, P., Milanesi, L. NuChart: an R package to study gene spatial neighbourhoods with multi-omics annotations. PLoS One, 2013, 8(9), e75146.
11. Deb, K., Jain, H. An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, Part I: Solving problems with box constraints. IEEE Transactions on Evolutionary Computation, 2014, 18(4):577–601.
12. 1000 Genomes Project Consortium. A global reference for human genetic variation. Nature, 2015, 526(7571):68–74.