AUT Journal of Electrical Engineering

AUT J. Elec. Eng., 55(2) (2023) 191-206
DOI: 10.22060/eej.2023.22434.5543

A rapid heuristic algorithm to solve the single individual haplotype assembly problem

M. Bagher, R. Karimzadeh, M. Jahed*, B. H. Khalaj

Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran

ABSTRACT: The Haplotype Assembly is the computational process in which two distinct nucleotide sequences of chromosomes are reconstructed using the sequencing reads of an individual. The ability to identify haplotypes provides many benefits for future genomic-based studies to be conducted in many areas, such as drug design, population study, and disease diagnosis. Even though several approaches have been put out to achieve highly accurate haplotypes, the problem of quick and precise haplotype assembly remains a challenging task. Due to the enormous bulk of the high-throughput sequencing data, algorithm speed plays a crucial role in the possibility of haplotype assembly in the human genome dimension. This study introduces a heuristic technique that enables rapid haplotype reconstruction while maintaining respectable accuracy. Our approach is divided into two parts. In the first, a partial haplotype is created and enlarged over a number of iterations. We have employed a novel metric to assess the reconstructed haplotype's quality in each iteration to arrive at the optimal answer. The second stage of the algorithm involves refining the reconstructed haplotypes to increase their accuracy. The outcome reveals that the suggested approach is capable of reconstructing the haplotypes with an acceptable level of accuracy. In terms of speed, the performance of the algorithm surpasses the competing approaches, especially in the case of high-coverage sequencing data.

Review History:

Received: Jun. 19, 2023
Revised: Aug. 07, 2023
Accepted: Aug. 08, 2023
Available Online: Oct. 01, 2023

Keywords:

Haplotype reconstruction
single nucleotide polymorphism
haplotype assembly
fragment
sequencing

1- Introduction

The genome of humans and other diploid species contains two homologous sets of chromosomes: a mother-inherited set and a father-inherited set. It has been discovered that the genome sequences of every two people are 99.5 percent identical, and variants in less than 1% of the genome play a major role in phenotypic variation and human disease [1]. Single nucleotide polymorphisms (SNPs) are sites in genome sequence where one DNA base differs between individuals [1, 2]. The term haplotype refers to a group of SNPs located on a distinct chromosome. The determination of haplotypes offers many advantages for genomic-based studies, like medical research, drug design, disease diagnosis, evolution studies, and population studies [3]. It is highly preferred to use computer algorithms to infer haplotypes since experimental methods are labor-intensive and time-consuming. Although this study focuses on diploid species, some groups of amphibians and food plants are included in the category of polyploid species, whose haplotype reconstruction has recently been considered in several articles [4, 5].

Due to the inability of current sequencing technologies to read an entire chromosome consecutively, the input data for the computational haplotype assembly problem are a collection of numerous short reads of DNA fragments. As each fragment is derived from one of the unknown haplotypes, the haplotype assembly task in diploids is to divide all these fragments into two subsets and obtain the corresponding two haplotypes. Under the assumption that all reads are noiseless, we can assemble the two original sequences by exploiting the overlaps between fragments. However, in practice, gaps and errors make this problem more challenging and complicated. A unique and correct genome assembly is guaranteed by two necessary and sufficient conditions for noiseless data. Firstly, the SNP coverage condition indicates that every SNP position should be covered by at least one fragment from each chromosome. Next, the Bridging condition states that every identical region between two chromosomes must be bridged by at least one fragment from either of the chromosomes [6]. Consequently, long-read data are needed to solve the haplotype assembly problem. As sequencing technologies advance, longer reads will be generated. There have already been several hundred kilobases read lengths achieved by third-generation sequencing technologies, including PacBio (Pacific Bio-Sciences) [7, 8] and ONT (Oxford Nanopore Technology) [9], While the largest fragments produced by second-generation sequencing technology are just 400 base pairs long [10]. However, lengthening fragments results in additional reading sequence errors. To lessen the impact of

*Corresponding author's email: jahed@sharif.edu

mistakes in haplotype assembly, it is critical to enhance data coverage. The requirement for fast algorithms is highlighted by this enormous quantity of fragments and the length of the human genome sequence. There are several models used for this purpose, the most well-known of which are Minimum SNP Removal (MSR) [11], Minimum Fragment Removal (MFR) [11], and Minimum Error Correction (MEC) [12]. In several recent algorithms [13-16], MEC has been applied as the most complex but most efficient model. This model seeks to divide the collection of fragments into two subsets with the fewest changes to the SNP values. Due to the NP-Hard nature of MEC and other models, some solutions take advantage of heuristic techniques by exchanging MEC score for rapidity. There are also several other proposed algorithms that have recently adopted strategies, such as matrix completion, to solve the haplotype assembly problem [17-19].

One of the earliest rapid heuristic methods to solve the haplotype assembly problem was the FastHare algorithm, presented by Panconesi and Sozio [20]. In this approach, the first aligned fragment is used to rebuild two partial haplotypes, and further fragments are allocated to one of the haplotypes depending on a given distance. There are a number of methods that make use of graphs with fragments as their vertices and distances or similarities between fragments as their edges. In other words, a measure like a similarity needs to be calculated between each pair of fragments, and the calculation cost relies on the number of reads. SSK, introduced by Xin-Shun Xu and Ying-Xin Li [21], is an instance of these methods. The first phase of this semi-supervised clustering method involves computing the similarity between each pair of fragments. Afterward, phase 2 utilizes some beneficial results gathered from phase 1 to aid k-means in clustering all the reads. A further heuristic technique called Fasthap [22] uses a fuzzy conflict graph of reads to create a preliminary split of the fragments according to differences between every two fragments; later, it refines the original grouping to improve the MEC score. Several additional methods are based on the fuzzy conflict graph that Fasthap first developed [23, 24]. As an example, FCMHap [23] infers two haplotypes from a fuzzy conflict graph and utilizes them as the starting centers of clustering for fuzzy c-means (FCM). A great deal of time is spent creating such a graph in the aforementioned approaches and any other novel methods that utilize a pairwise distance between fragments.

In this study, we present a heuristic algorithm that achieves a high speed of haplotype assembly while maintaining an acceptable MEC score. In contrast to the previous methods, this method does not require calculating the distance between all pairs of fragments, but partial haplotypes are constructed based on a large number of fragments in each iteration, so it accelerates convergence and brings about a reduction in computational cost.

This paper has a structure as follows: Section 2 presents a brief definition of the haplotype assembly problem, followed by a discussion of the problem formulation, our approach, and evaluation metrics. In section 3, a description of the dataset and materials is provided, and then the results are discussed. Lastly, the conclusion of the study is presented in section 4.

2- Materials and Methods

2- 1- Problem definition

As mentioned before, a set of reads provides the input to the haplotype assembly problem. A sequence aligner is used to map the reads to a reference genome before beginning any haplotype assembly method. Once the loci of heterozygous variants are identified, they are written into an m × n variant matrix, where m and n are the numbers of reads and SNPs, respectively. The haplotype assembly task involves clustering reads into two groups and determining the consensus sequences for each group to reconstruct the haplotypes. Fig. 1 illustrates how a haplotype can be reconstructed from noiseless fragments.

It is currently impossible for any of the high throughput sequencing technologies to produce error-free reads; therefore, the corresponding SNP matrix would not be bipartisan. It is necessary to consider some components of the SNP matrix to be errors and flip them to produce a bipartisan matrix. To evaluate the efficiency of different algorithms, the number of these error corrections is reported as the MEC measure. As previously stated, attaining the minimum of this metric is an NP-hard task; hence heuristic methods are preferable.

2- 2- Formulation and proposed method

Suppose M is an input SNP fragment matrix of size $m \times n$, where the rows represent fragments, the columns represent SNP sites, and its values are $m_{ij} \in \{0,1,-1\}$. In this study, each matrix member m_{ij} is assigned one of the integers 1, -1 or 0, depending on whether the i^{th} site of the j^{th} read corresponds to the most frequent allele, the rare one in the i^{th} column, or a gap, respectively. The procedure of constructing matrix M is depicted in Fig. 2. An approximate haplotype is represented by $\hat{H} = \{\hat{h}_1, \hat{h}_2\}$, where $\hat{h}_{k(1\times n)}$ is a vector in which $\hat{h}_{kj} \in \{1,-1\}$ and the subscript k is assigned the values 1 or 2, indicating whether $\hat{h}_k$ is the first or second haplotype. $\hat{h}_1$ and $\hat{h}_2$ are randomly initialized with the condition that $\hat{h}_{1j} = -\hat{h}_{2j}$. To measure the compatibility between reads and each estimated haplotype $\hat{h}_k$, we have created a vector namely fragment compatibility (*fc*) by:

$$fc_k = M \times \hat{h}_k^{\,T} = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_m \end{bmatrix} \quad (1)$$

Where δ_i is the i^{th} component of fc and indicates the compatibility between $f_i = \{m_{i1}, m_{i2}; ..., m_{in}\}$ and determined the haplotype $\hat{h}_k$ as follows:

$$\delta_i = \sum_{j=1}^{n} m_{ij}.\hat{h}_k = f_i.\hat{h}_k \quad (2)$$

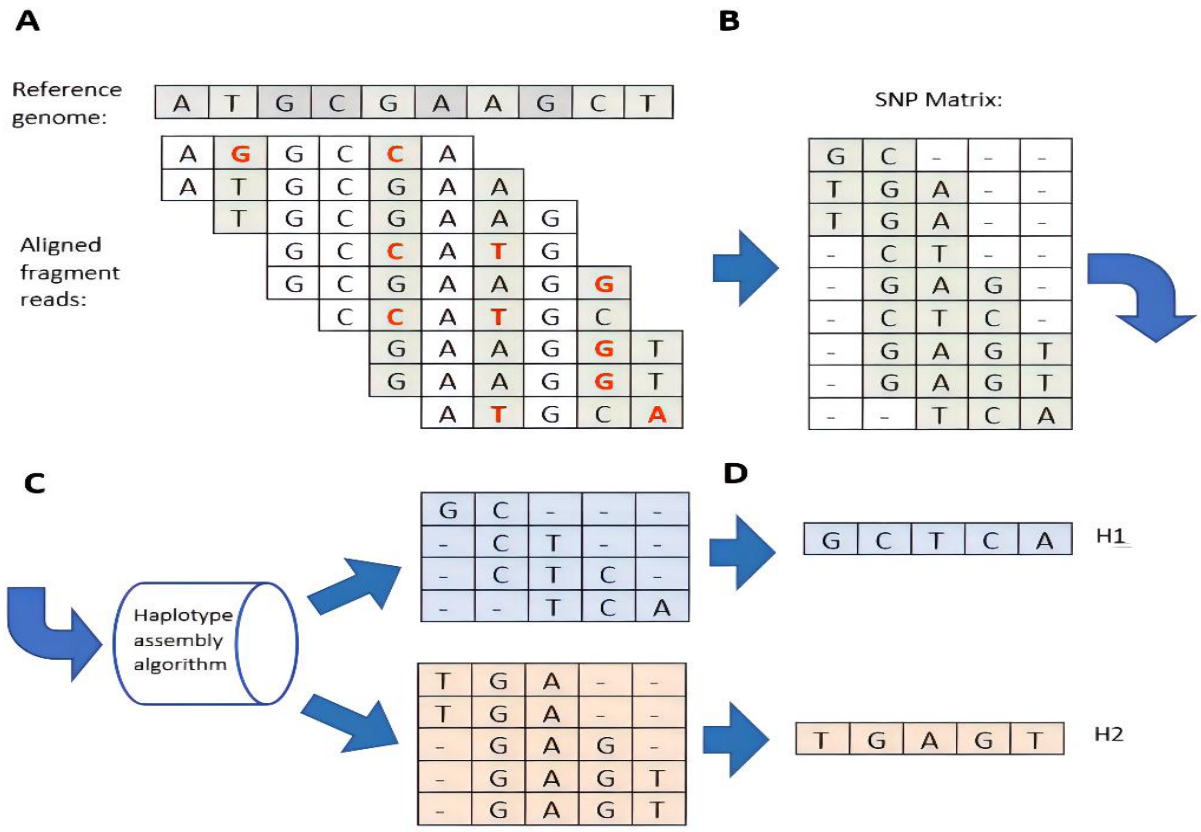

Fig. 1. A demonstration of a haplotype assembly process using error-free fragments

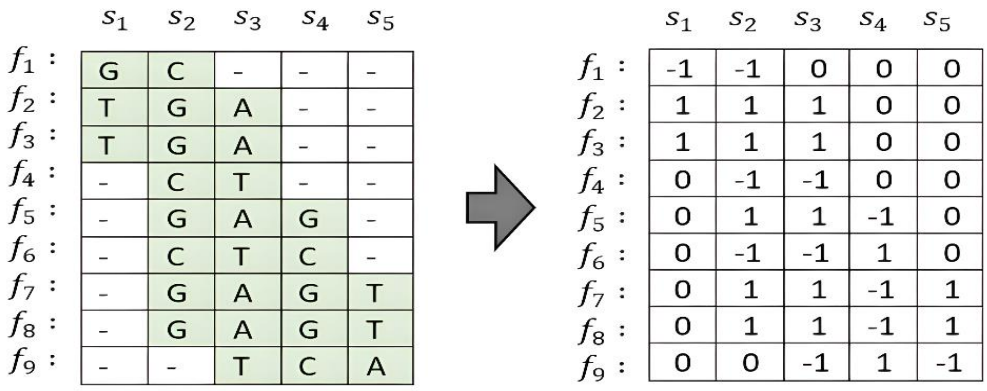

Fig. 2. The procedure of constructing the numeric matrix from SNP fragments

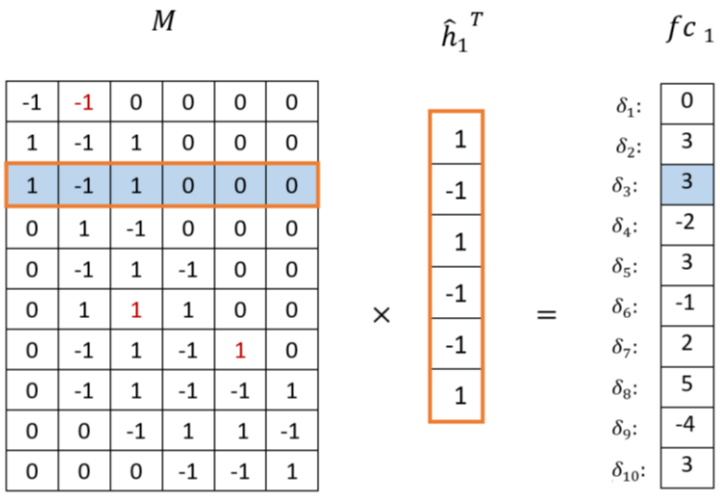

Fig. 3. An example of calculating fc from the estimated h^_1 in a simple noisy fragment matrix

In Fig. 3 an example has been provided that depicts Eq.1 and 2 in a practical context. When $|\delta_i|$ (absolute value of δ_i) is large, it indicates that the fragment $\boldsymbol{f}_i$ has a high probability of being compatible with either of the two determined haplotypes, which makes it easier to cluster. In other words, as δ_i becomes more positive, the probability that $\boldsymbol{f}_i$ will be related to $\hat{h}_1$ is increased. On the other hand, as δ_i becomes more negative, the probability that $\boldsymbol{f}_i$ will be related to another haplotype $\hat{h}_2$ is increased, and therefore it can be deduced that $\hat{h}_{1j} = -\hat{h}_{2j}$. In the case of a zero value of δ_i, you can consider the problem as being a case that $\boldsymbol{f}_i$ cannot be grouped into either $\hat{\boldsymbol{h}}_1$ or $\hat{\boldsymbol{h}}_2$.

Our presented algorithm, which we've called QuickHap, comprises two main phases: the first is the quick partitioning stage, and the second is the refinement stage. The flowchart in Fig. 4 depicts the different stages of the algorithm in great detail. The first phase begins with the construction of an approximate preliminary haplotype, which is then iteratively expanded utilizing a lot of fragments at each cycle. To put it another way, in each iteration, a lot of fragments are clustered using the fragment compatibility vector. In this paper, we present a new metric, Clustering Rate (CR), which assesses the validity of the reassembled haplotype and is defined as follows:

$$CR_k = \frac{\sum_{i=1}^{m} \left| \sum_{j=1}^{n} m_{ij}\, \hat{h}_{k_j} \right|}{\sum_{i=1}^{m} \sum_{j=1}^{n} |m_{ij}|^2} = \frac{\left\| M \times \hat{h}_k{}^T \right\|_1}{\|M\|_F^2} \quad (3)$$

$$CR = \frac{CR_1 + CR_2}{2} \quad (4)$$

This measure always returns a number between 0 and 1. If haplotypes $\hat{h}_1$ and $\hat{h}_2$ are defined in such a way that all fragments fit well into one of the two haplotypes, this number will be closer to 1. In the case of noiseless data, when haplotypes are accurately determined, this number attains 1. As the fragments and haplotypes become further incompatible, that is, as fragments do not fit any of the two haplotypes, this value will tend towards 0.

The first step of the process is to select the fragment with the most compatibility value with the starting h1 to be used as the new h1. Every iteration involves a clustering process where fragments with positive compatibility value are gathered into C1 (group of h1-related fragments), and fragments with negative value are clustered into C2 (group of h2-related fragments). Thereafter, for each cluster C, members are used to create related h_new as follows:

$$h\_new_j = \begin{cases} \dfrac{\sum_i c_{ij}}{\left|\sum_i c_{ij}\right|} & \sum_i c_{ij} \neq 0 \\ 0 & otherwise \end{cases} \quad (5)$$

If we assume C1 and C2 as two newly separated matrices, for each matrix, c_{ij} is the j^{th} SNP of i^{th} fragment belonging to Cluster C and h_new_j is the j^{th} SNP of the related haplotype h_new. Eq. 5 is a voting process that can be rewritten as below:

$$h\_new_j = \begin{cases} +1 & n_1(j) > n_{-1}(j) \\ -1 & n_1(j) < n_{-1}(j) \\ 0 & otherwise \end{cases} \quad (6)$$

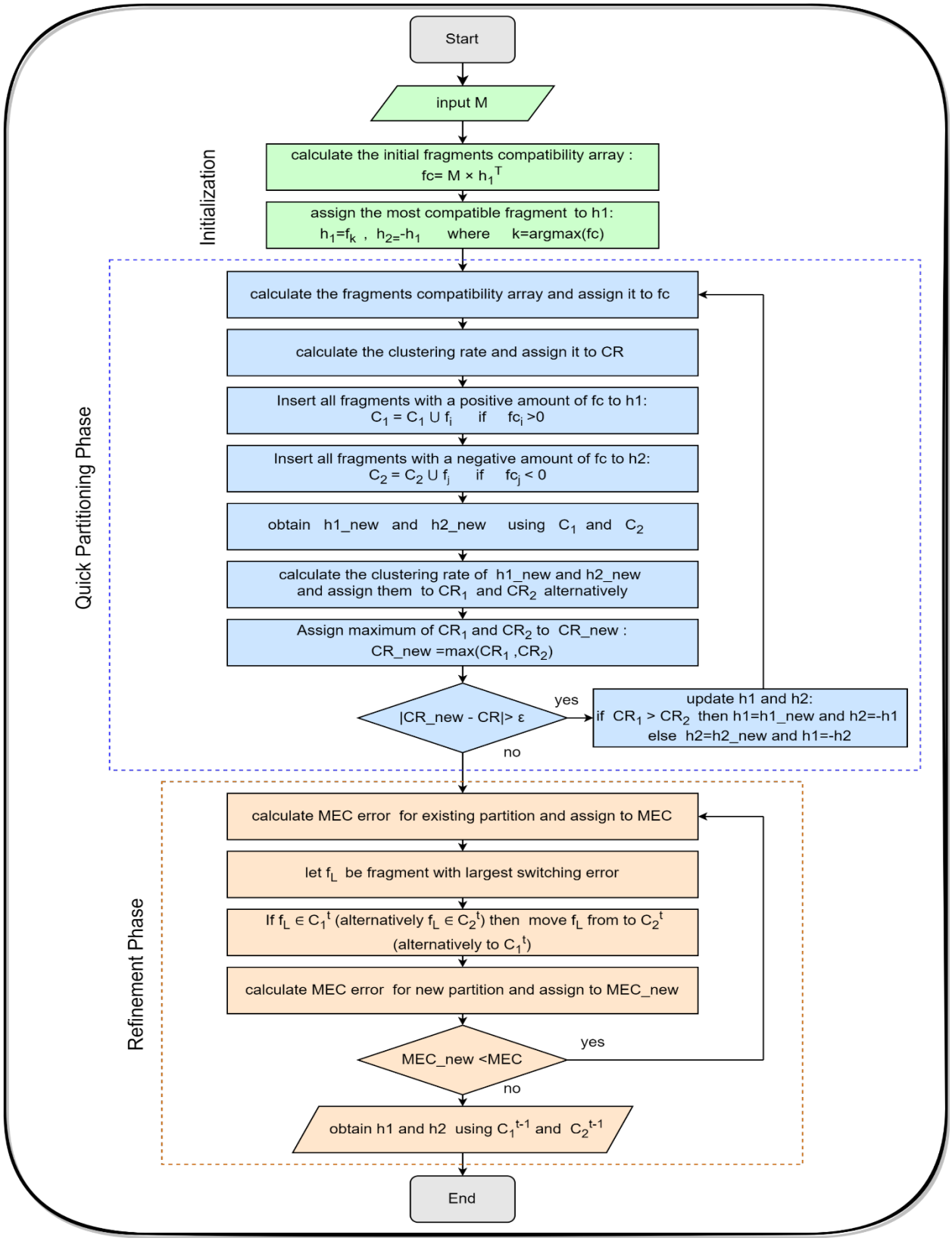

Fig. 4. Flowchart of the proposed approach

Where $n_1(j)$ is the number of fragments whose j^{th} element is 1, and $n_{-1}(j)$ represents the number of fragments with j^{th} SNP of -1. As a final step of phase 1, The CR of both $h1\_new$ and $h2\_new$ are recalculated; if the enhancement in CR is considerable, the entire procedure is repeated.

Two clustered groups of fragments from the first stage's output are used as the second stage's input. The second stage is carried out in accordance with the Fasthap method's refinement phase; MEC Scores are calculated for approximated clusters and haplotypes, and the fragments with the greatest variations within their group are shifted to the other set. This phase is repeated as long as a decrease in the MEC error occurs during the process. A formula for calculating MEC is provided below in the section devoted to evaluation.

2- 3- Performance evaluation

To assess the performance of the algorithm, A number of measurements have been computed, including MEC, the execution time, and the Reconstruction Rate (RR). As previously stated, the MEC score refers to the minimum number of SNP alterations necessary to divide the variation matrix into two haplotype-specific submatrices. The following formula is used to obtain the MEC score:

$$d(x,y)=\begin{cases}1 & if \quad x\neq y \quad \& \, x,y \in \{1,-1\} \\ 0 & otherwise\end{cases} \tag{7}$$

$$HD(f,h)=\sum_{j=1}^{n} d(f_j,h_j) \tag{8}$$

$$MEC=\sum_{i=1}^{m} min\{HD(f_i,\hat{h}_1),HD(f_i,\hat{h}_2)\} \tag{9}$$

The RR will be helpful if the actual haplotypes are known, and the ground truth is accessible. In our research, the real haplotypes were accessible to us via the data simulation. If $H=\{h_1,h_2\}$ are the accurate haplotypes, the correctness of the approximated haplotypes $\hat{H}=\{\hat{h}_1,\hat{h}_2\}$ would be described by RR as follows:

$$RR=1-\frac{\min\{HD(h_1,\hat{h}_1)+HD(h_2,\hat{h}_2),HD(h_1,\hat{h}_2)+HD(h_2,\hat{h}_1)\}}{2n} \tag{10}$$

3- Validation.

3- 1- Dataset and setup

The read sequences have been simulated utilizing PBSIM, which is a simulator for PacBio sequencing reads [25, 26]. The synthetic (yet realistic) sequencing reads were derived from the APP gene of chromosome 21 of the GRCh38 reference genome [27], and random SNPs were inserted. From the output FASTQ files generated as a result of PBSIM, the BAM and VCF files were extracted, and fragment files were generated using the ExtractHairs program [28, 29]. Simulations were performed using various parameter combinations such as coverage $C=\{5,10,20\}$, read length $L=\{5,7.5,10,15,20,25\}$ kbp, and error rate $e=\{0.01,0.05,0.1,0.2,0.3,0.4\}$. The experiments were conducted on an Intel Core i7 processor running 2.6 GHz with 16GB of RAM.

3- 2- Results

QuickHAP is compared in this experiment with several state-of-the-art methods, including SSK, FastHAP, and FCMHAP, which are based on fuzzy conflict graphs of fragments. The impact of error rate on the efficiency of various algorithms is depicted in Fig. 5. In this graph, the average MEC Score for every method is plotted against its error rate. The MEC cost of our approach QuickHap, as can be seen, is below the MEC of SSK and FCMHap by a considerable margin for all error rates and engages in rivalry with the FASTHap method.

The average MEC costs for the algorithms are plotted versus coverage and read length in Figs. 6 and 7, respectively, providing a comparison in which, just like in Fig. 5, FastHap is the only algorithm to compete with QuickHap.

Since there is no ground truth in the haplotype assembly problem on the actual data, the MEC criterion is considered as the most useful measure for comparing the performance of the haplotype assembly algorithms. Still, since it has been proven in some articles that a lower MEC criterion does not necessarily mean a higher accuracy [30], another measure, namely RR is evaluated in our study, which is mainly applicable to the simulated data.

In Figures 8, 9, and 10, the average RRs are plotted against the error rate, coverage, and read length, respectively. The bar graphs reveal that QuickHap, FCMHap, and FastHap are competing, particularly with increasing error rates.

The average time of execution for various error rates, coverages, and read lengths are presented in Fig. 11, 12, and 13, respectively. This study demonstrates that the algorithms' execution times get shorter as the read length grows; this is because longer reads of data with fixed coverage result in fewer fragments and faster analysis. It is worth to note that in all graphs, QuickHAP's average execution time is shorter than that of the other three techniques.

By taking a closer look at the bar charts, it can be concluded that when measuring method performance based on MEC cost, FastHap and QuickHap are in competition, and if Reconstruction Rate is taken into account as a criterion of accuracy, FCMHap, FastHAP, and QuickHap are roughly on par. Additionally, in our proposed approach, we have observed an improvement in average execution time almost across all error rates, coverages, and read lengths. To allow for a more detailed analysis of the result, Tables 1-3 are provided for data with a read length of 5000.

The MEC cost of algorithms across different coverage and error rate combinations are listed in Table 1, and the best

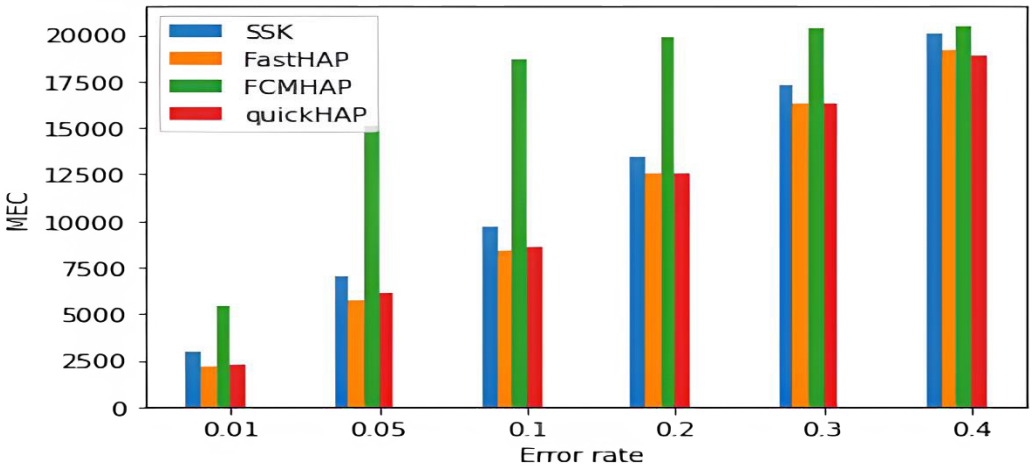

Fig. 5. The average MEC costs of the algorithms versus error rate

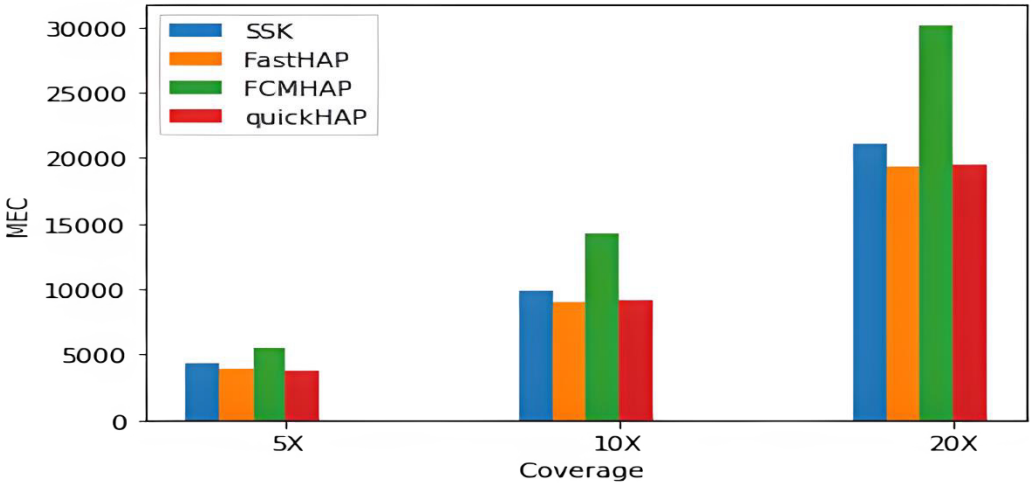

Fig. 6. The average MEC costs of the algorithms versus coverage

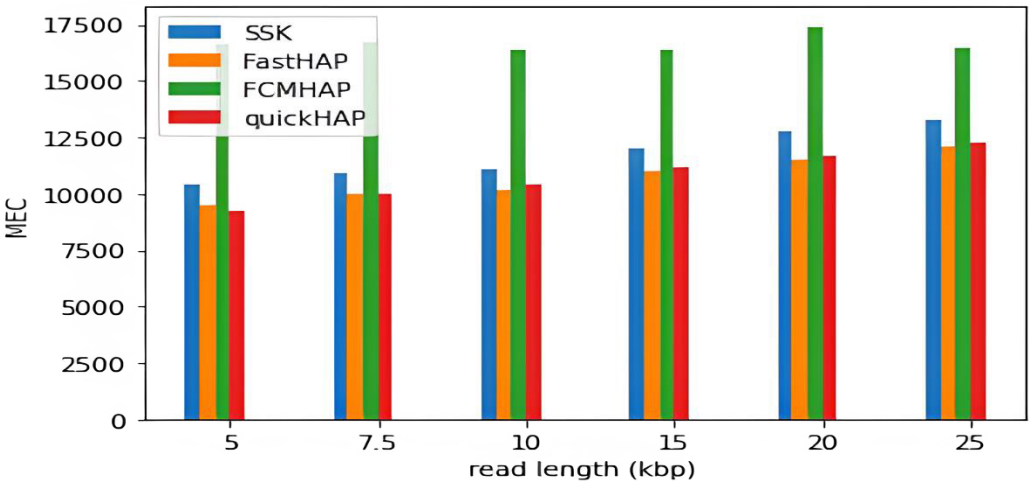

Fig. 7. The average MEC cost of the algorithms versus read length

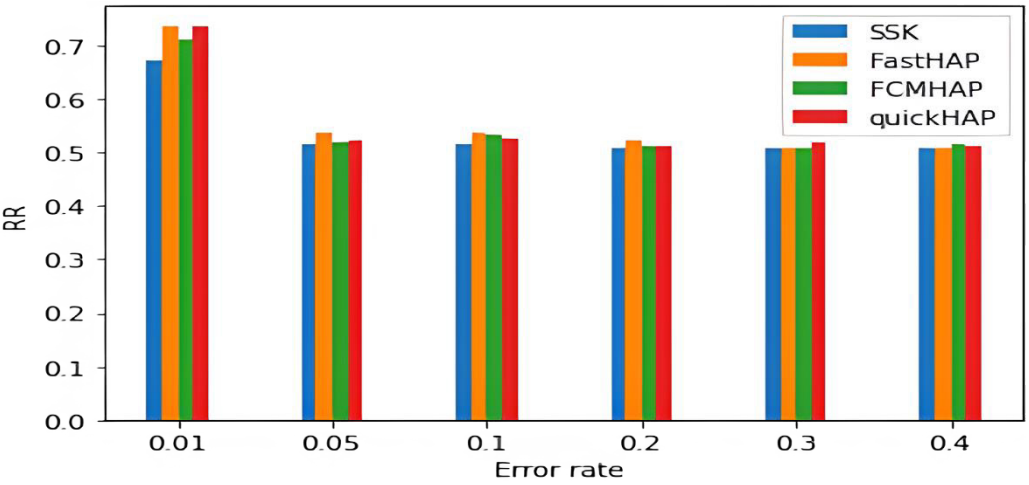

Fig. 8. Average RR of the algorithms versus error rate

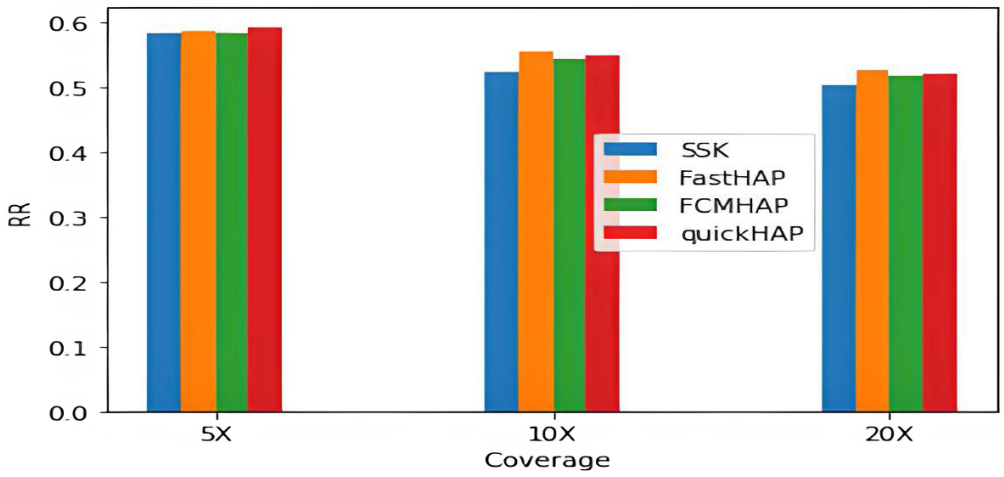

Fig. 9. Average RR of the algorithms versus coverage

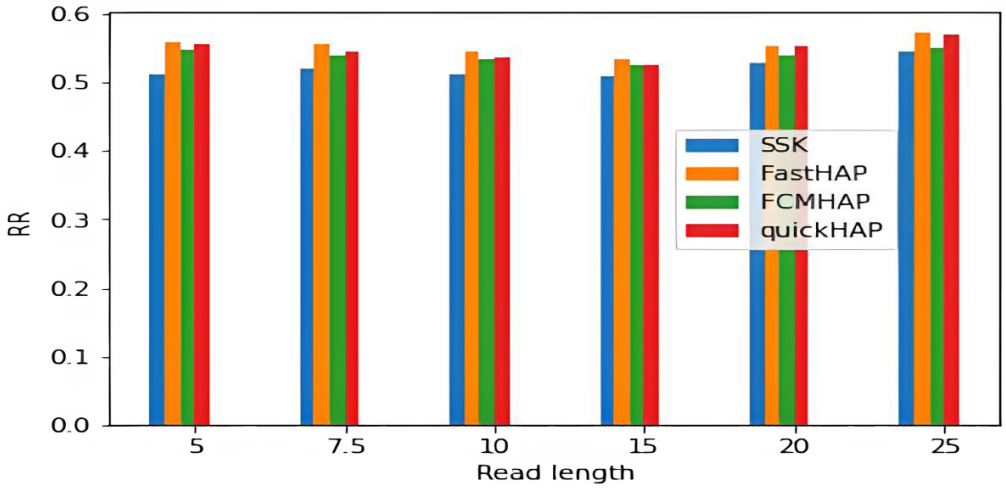

Fig. 10. Average RR of the algorithms versus read length

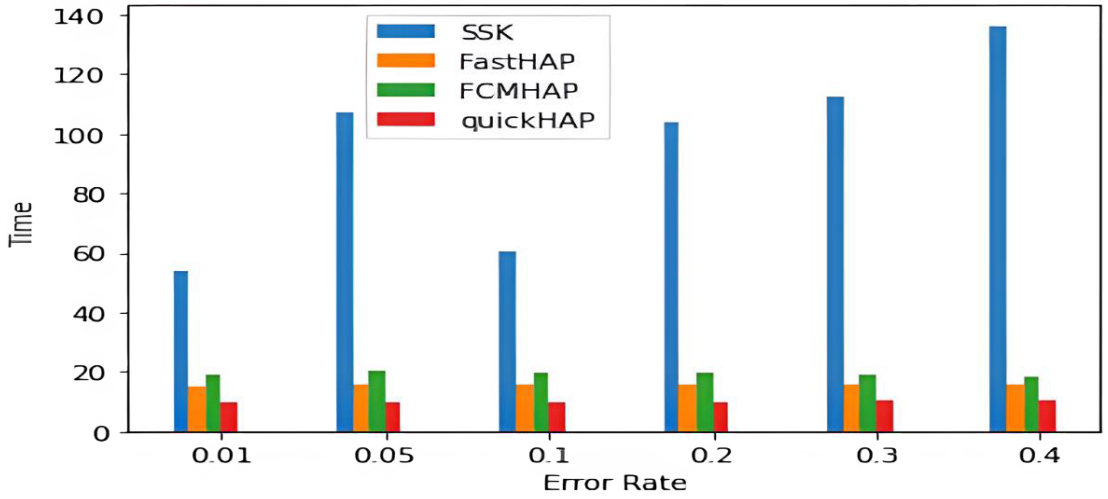

Fig. 11. Average execution time of the algorithms versus error rate

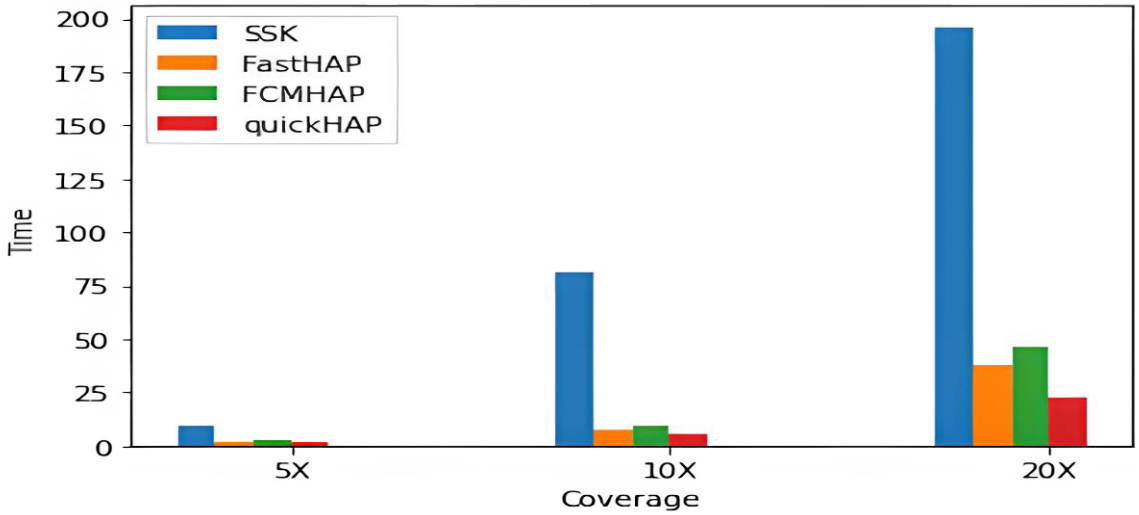

Fig. 12. Average execution time of the algorithms versus coverage

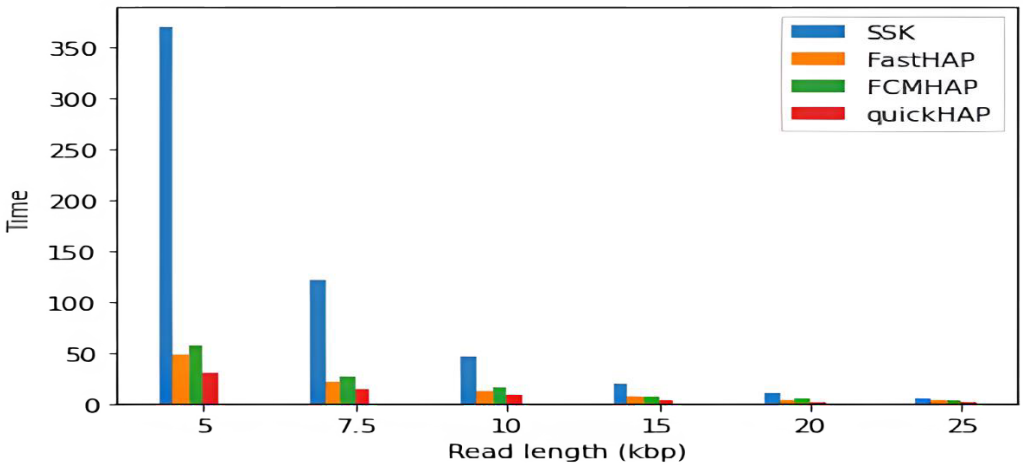

Fig. 13. Average execution time of the algorithms versus read length

Table 1. A comparison of MEC costs based on coverage and error rate

C	*e*	*SSK*	*FastHap*	*FCMHap*	*Quickhap*
5	*0.01*	*424*	*456*	*1280*	***390***
5	*0.05*	*1904*	*1484*	*4083*	***1476***
5	*0.1*	*3340*	*2663*	*5453*	***2637***
5	*0.2*	*4899*	***4476***	*6936*	*4493*
5	*0.3*	*6309*	***5995***	*7323*	*6202*
5	*0.4*	*7029*	***6720***	*7327*	*6982*
10	*0.01*	*1240*	***870***	*4350*	***870***
10	*0.05*	*3828*	***2999***	*11748*	*3129*
10	*0.1*	*6543*	***5193***	*14572*	*5360*
10	*0.2*	*10878*	*9495*	*16105*	***9450***
10	*0.3*	*13882*	***13361***	*16720*	*13977*
10	*0.4*	*16102*	***15621***	*16764*	*15733*
20	*0.01*	*3161*	***2069***	*14782*	*2304*
20	*0.05*	*9228*	***6923***	*29837*	*7494*
20	*0.1*	*13605*	***11497***	*33967*	*12003*
20	*0.2*	*21241*	***19352***	*35547*	*20323*
20	*0.3*	*29726*	***28079***	*35971*	*28792*
20	*0.4*	*34856*	***33807***	*36188*	*34022*

Table 2. A comparison of RR based on coverage and error rate

C	*e*	*SSK*	*FastHap*	*FCMHap*	*Quickhap*
5	*0.01*	*0.9142*	*0.9178*	*0.8916*	***0.9220***
5	*0.05*	*0.5234*	***0.5345***	*0.5136*	*0.5187*
5	*0.1*	*0.5317*	*0.5317*	*0.5438*	***0.5638***
5	*0.2*	*0.5069*	*0.5163*	*0.5069*	***0.5228***
5	*0.3*	*0.5139*	*0.5130*	***0.5191***	*0.5139*
5	*0.4*	*0.5170*	*0.5142*	***0.5231***	*0.5198*
10	*0.01*	*0.5989*	***0.7085***	*0.6659*	*0.7076*
10	*0.05*	*0.5145*	***0.5426***	*0.5255*	*0.5357*
10	*0.1*	*0.5050*	***0.5554***	*0.5415*	*0.5073*
10	*0.2*	*0.5118*	***0.5210***	*0.5169*	*0.5104*
10	*0.3*	*0.5055*	*0.5*	*0.5046*	***0.5143***
10	*0.4*	*0.5018*	*0.5051*	*0.5125*	***0.5166***
20	*0.01*	*0.5002*	***0.5794***	*0.5744*	*0.5785*
20	*0.05*	*0.5078*	***0.5309***	*0.5133*	*0.5124*
20	*0.1*	*0.5069*	***0.5185***	*0.5087*	*0.5050*
20	*0.2*	*0.5018*	***0.5254***	*0.5064*	*0.5013*
20	*0.3*	*0.5023*	*0.5092*	*0.5041*	***0.5217***
20	*0.4*	*0.5027*	*0.5041*	***0.5101***	*0.5018*

outcomes in each row are highlighted in bold.

Table 2 details the average RRs of the algorithms in various error rates and coverages. The highest value in each row is highlighted in bold. As can be seen, QuickHap frequently competes with FastHAP and FCMHap, sometimes even surpassing them.

Table 3 demonstrates the details of the algorithm's performance in terms of speed for different coverages and error rates. According to the running time values, QuickHap is the most rapid algorithm in almost all cases. This superiority in speed is particularly more evident in data with high coverage.

According to the obtained results, the proposed algorithm can solve the haplotype reconstruction problem for almost all inputs with the highest speed and acceptable accuracy. As stated earlier, SSK, FastHap, and FCMHap are based on a distance calculation between each pair of reads. Therefore, these methods have the drawback of slowing down the reconstruction process when increasing coverage and the number of fragments. Based on the results of our speed analysis, we can conclude that the proposed algorithm is capable of reconstructing haplotypes at a promising speed, particularly when dealing with high-coverage sequencing data.

Table 3. A comparison of execution time based on coverage and error rate

C	e	SSK	FastHap	FCMHap	Quickhap
5	0.01	5.702	5.160	6.289	**5.101**
5	0.05	7.405	5.018	5.960	**4.933**
5	0.1	9.050	5.215	5.995	**5.073**
5	0.2	9.534	5.367	6.008	**5.207**
5	0.3	10.219	**5.654**	5.983	6.014
5	0.4	10.567	**5.819**	6.165	6.424
10	0.01	31.495	23.911	24.689	**20.232**
10	0.05	85.297	25.577	25.273	**18.259**
10	0.1	39.634	23.891	26.011	**17.260**
10	0.2	55.638	24.321	26.150	**18.264**
10	0.3	81.342	23.950	24.773	**21.105**
10	0.4	82.946	**22.854**	24.051	23.769
20	0.01	152.731	115.581	124.170	**71.577**
20	0.05	195.993	123.382	124.918	**66.778**
20	0.1	162.282	126.666	127.922	**70.840**
20	0.2	430.195	126.057	132.084	**73.871**
20	0.3	303.234	125.017	125.920	**87.225**
20	0.4	554.48	112.716	116.032	**92.974**

4- Conclusion

This work presented a heuristic algorithm for solving the haplotype assembly problem in diplids, consisting of two phases, a quick partitioning phase, and a refinement phase. The first phase involved defining a fragment compatibility vector (fc) and introducing the Clustering Rate (CR) as a new criterion. The proposed approach utilized an iterative process in which many fragments are clustered using fc in each iteration, and a partial haplotype was gradually expanded to maximize CR. During the second phase, reconstructed haplotypes were refined to obtain the lowest MEC score possible. The results of this study demonstrate that the proposed method is capable of reconstructing haplotypes with a high degree of speed and accuracy, particularly for high-coverage sequencing data.

References

[1] P. Gupta, J. Roy, M. Prasad, Single nucleotide polymorphisms: a new paradigm for molecular marker technology and DNA polymorphism detection with emphasis on their use in plants, Current science, (2001) 524-535.

[2] R.A. Gibbs, J.W. Belmont, P. Hardenbol, T.D. Willis, F. Yu, H. Yang, L.-Y. Ch'ang, W. Huang, B. Liu, Y. Shen, The international HapMap project, (2003).

[3] X. Zhu, S. Zhang, D. Kan, R. Cooper, Haplotype block definition and its application, in: Biocomputing 2004, World Scientific, 2003, pp. 152-163.

[4] M.-H. Moeinzadeh, J. Yang, E. Muzychenko, G. Gallone, D. Heller, K. Reinert, S. Haas, M. Vingron, Ranbow: a fast and accurate method for polyploid haplotype reconstruction, PLOS Computational Biology, 16(5) (2020) e1007843.

[5] S. Majidian, M.H. Kahaei, D. De Ridder, Hap10: Reconstructing accurate and long polyploid haplotypes using linked reads, BMC bioinformatics, 21(1) (2020) 1-18.

[6] A. Najafi, D. Nashta-ali, S.A. Motahari, M. Khani, B.H. Khalaj, H.R. Rabiee, Fundamental limits of pooled-DNA sequencing, arXiv preprint arXiv:1604.04735, (2016).

[7] A. Rhoads, K.F. Au, PacBio sequencing and its applications, Genomics, proteomics & bioinformatics, 13(5) (2015) 278-289.

[8] R.J. Roberts, M.O. Carneiro, M.C. Schatz, The advantages of SMRT sequencing, Genome biology, 14(6) (2013) 1-4.

[9] H. Lu, F. Giordano, Z. Ning, Oxford Nanopore MinION sequencing and genome assembly, Genomics, proteomics & bioinformatics, 14(5) (2016) 265-279.

[10] M.A. Quail, I. Kozarewa, F. Smith, A. Scally, P.J. Stephens, R. Durbin, H. Swerdlow, D.J. Turner, A large genome center's improvements to the Illumina sequencing system, Nature methods, 5(12) (2008) 1005-1010.

[11] G. Lancia, V. Bafna, S. Istrail, R. Lippert, R. Schwartz, SNPs problems, complexity, and algorithms, in: ESA, Springer, 2001, pp. 182-193.

[12] R. Lippert, R. Schwartz, G. Lancia, S. Istrail, Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem, Briefings in bioinformatics, 3(1) (2002) 23-31.

[13] V. Bansal, V. Bafna, HapCUT: an efficient and accurate algorithm for the haplotype assembly problem, Bioinformatics, 24(16) (2008) i153-i159.

[14] W. Qian, Y. Yang, N. Yang, C. Li, Particle swarm optimization for SNP haplotype reconstruction problem, Applied mathematics and Computation, 196(1) (2008) 266-272.

[15] T.-C. Wang, J. Taheri, A.Y. Zomaya, Using genetic algorithm in reconstructing single individual haplotype with minimum error correction, Journal of biomedical informatics, 45(5) (2012) 922-930.

[16] M.-H. Olyaee, A. Khanteymoori, AROHap: An effective algorithm for single individual haplotype reconstruction based on asexual reproduction optimization, Computational biology and chemistry, 72 (2018) 1-10.

[17] S. Majidian, M.H. Kahaei, NGS based haplotype assembly using matrix completion, PLoS One, 14(3) (2019) e0214455.

[18] S. Majidian, M.M. Mohades, M.H. Kahaei, Matrix completion with weighted constraint for haplotype estimation, Digital Signal Processing, 108 (2021) 102880.

[19] M.M. Mohades, S. Majidian, M.H. Kahaei, Haplotype assembly using manifold optimization and error correction mechanism, IEEE Signal Processing Letters, 26(6) (2019) 868-872.

[20] A. Panconesi, M. Sozio, Fast hare: A fast heuristic for single individual SNP haplotype reconstruction, in: Algorithms in Bioinformatics: 4th International Workshop, WABI 2004, Bergen, Norway, September 17-21, 2004. Proceedings 4, Springer, 2004, pp. 266-277.

[21] X.-S. Xu, Y.-X. Li, Semi-supervised clustering algorithm for haplotype assembly problem based on MEC model, International journal of data mining and bioinformatics, 6(4) (2012) 429-446.

[22] S. Mazrouee, W. Wang, FastHap: fast and accurate single individual haplotype reconstruction using fuzzy conflict graphs, Bioinformatics, 30(17) (2014) i371-i378.

[23] M.H. Olyaee, A. Khanteymoori, E. Fazli, A fuzzy c-means clustering approach for haplotype reconstruction based on minimum error correction, Informatics in Medicine Unlocked, 25 (2021) 100646.

[24] F. Zamani, M.H. Olyaee, A. Khanteymoori, NCMHap: a novel method for haplotype reconstruction based on Neutrosophic c-means clustering, Bmc Bioinformatics, 21 (2020) 1-15.

[25] Y. Ono, K. Asai, M. Hamada, PBSIM: PacBio reads simulator—toward accurate genome assembly, Bioinformatics, 29(1) (2013) 119-121.

[26] Y. Ono, K. Asai, M. Hamada, PBSIM2: a simulator for long-read sequencers with a novel generative model of quality scores, Bioinformatics, 37(5) (2021) 589-595.

[27] APP amyloid beta precursor protein [Homo sapiens (human)] - Gene - NCBI, in, National Center for Biotechnology Information.

[28] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, G.P.D.P. Subgroup, The sequence alignment/map format and SAMtools, bioinformatics, 25(16) (2009) 2078-2079.

[29] P. Edge, V. Bafna, V. Bansal, HapCUT2: robust and accurate haplotype assembly for diverse sequencing technologies, Genome research, 27(5) (2017) 801-812.

[30] S. Majidian, M.H. Kahaei, D. de Ridder, Minimum error correction-based haplotype assembly: Considerations for long read data, Plos one, 15(6) (2020) e0234470.

HOW TO CITE THIS ARTICLE

M. Bagher, R. Karimzadeh, M. Jahed, B. H. Khalaj, A rapid heuristic algorithm to solve the single individual haplotype assembly problem , AUT J. Elec. Eng., 55(2) (2023) 191-206.

DOI: 10.22060/eej.2023.22434.5543