# A Hybrid Approach to Optimize the Number of Recombinations in Ancestral Recombination Graphs

Nguyen Thi Phuong Thao
Institute of Information Technology
Vietnam Academy of Science and Technology
+84-9-1621-8689
thaontp@ioit.ac.vn

Le Sy Vinh
University of Technology and Engineering
Vietnam National University, Hanoi
+84-9-0226-2444
vinhls@vnu.edu.vn

## ABSTRACT
Building ancestral recombination graphs (ARG) with minimum number of recombination events for large datasets is a challenging problem. We have proposed ARG4WG and REARG heuristic algorithm for constructing ARGs with thousands of whole genome sequences. However, these algorithms do not result in ARGs with minimal number of recombination events. In this work, we propose GAMARG algorithm, an improvement of ARG4WG, to optimize the number of recombination events in ARG building process. Experiment with different datasets showed that GAMARG algorithm outperforms other heuristic algorithms in building ARGs for large datasets. It also is much better than other heuristic algorithms and comparable to exhaustive search methods for small datasets.

## CCS Concepts
**Applied computing → Life and medical sciences → Bioinformatics**

## Keywords
Ancestral recombination graphs; Minimal ARG; Minimum number of recombinations; Recombination breakpoint

## 1. INTRODUCTION
Ancestral recombination graph (ARG) plays a central role in the analysis of within-species genetic variations [1]. The relationships between current species and common ancestors can be described by coalescence, mutation and recombination events in the ARG (Figure 1). Looking backward in time, the coalescence events merge two identical sequences to one; the mutation events make change in a site of the sequence; the recombination events break one sequence to two subsequences that then make change in the genetic information of the next generations. So the mutation and recombination events are important factors in consideration when building ARGs.

Approaches have been proposed to infer ARGs. Most of methods use the infinite-sites assumption that does not allow back and recurrent mutation in a single site. Thus, they try to build ARGs with the minimum number of recombination events. This is proved an NP-hard problem [2].

Several methods have been proposed to construct ARGs with optimal number of recombination events (called minimal ARGs) for small datasets. Song et al. [3] built minimal ARGs by scanning all possible ways and selecting the best way to move trees for each marker from left to right along the sequence to optimize the number of recombination events. Given a number of recombination events, Lyngsø et al. [4] tried to construct an ARG using a branch and bound algorithm. If it is impossible to have an ARG, the number of recombination events is increased by one. The process is continued until an ARG is constructed. All these exhaustive search methods have very high computational complexity. They are just able to work with up to dozens of short sequences.

To deal with larger datasets, other heuristic methods have been proposed. In spite of focusing on building minimal ARGs, they try to build plausible ARGs. Margarita proposed by Minichiello and Durbin [5] can handle a thousand sequences with hundreds of markers; ARG4WG of our group [6] can handle thousands of whole human genomes. The longest shared ends criterion in building ARGs allows ARG4WG to work on large datasets with less number of recombination events than Margarita. Our experiments showed that ARG4WG is able to build ARG with fewer number of recombination events than Margarita but still does not reach the minimal ARGs.

To build large ARGs with minimum number of recombination events, we evaluated the effect of different factors on reducing the number of recombination events in ARG building process for genome-wide and suggested a new design of ARG4WG, called REARG [7]. Specifically, we combined some other factors such as similarity between sequences and the length of sequences into REARG. This strategy enables REARG to build ARGs with a smaller number of recombination events in comparison to ARG4WG. However, REARG still is not as good as other exhaustive search methods.

As the longest shared ends criterion does not result in the minimum number of recombination events (Figure 2b), we should combine ARG4WG with other optimal criteria to reduce the recombination events. Notably, the four-gamete test [8] is the key idea leading to various methods either to find the lower bound of the number of recombination events or to construct explicitly minimum recombination ARG. In this work, we propose GAMARG method to build large ARGs with the minimum number of recombination events. Our experiments on different datasets showed that GAMARG algorithm is able to handle thousands sequences with tens of thousands of markers, and also could reach the minimum recombination ARGs.


*ICBBB '19*, January 7–9, 2019, Singapore, Singapore

ACM ISBN 978-1-4503-6654-0/19/01$15.00
DOI: https://doi.org/10.1145/3314367.3314385

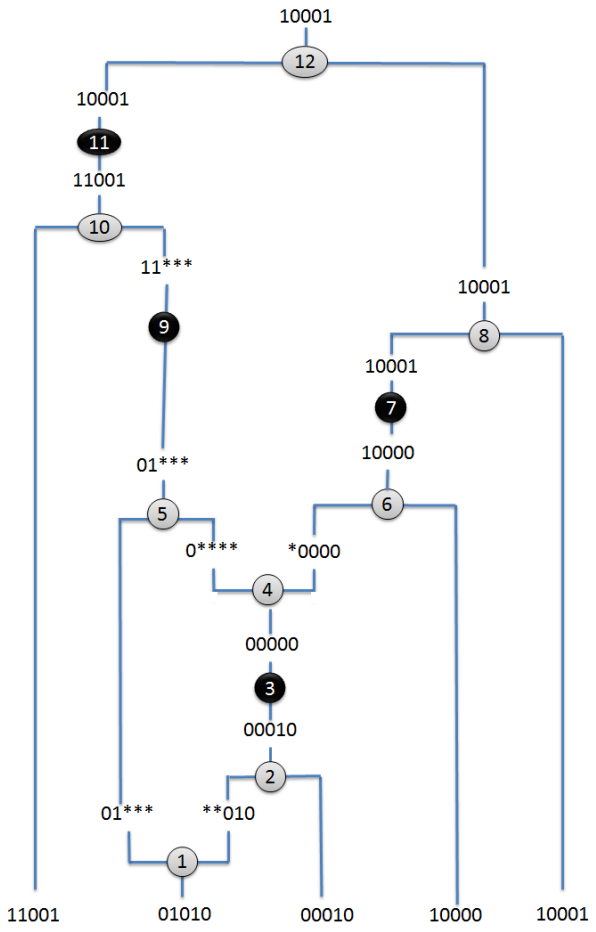



**Figure 1. An example of an ARG for 5 sequences of length 5**

The paper is organized as follows. In section II, we introduce the ARG building problem. Some problems in choosing the breakpoint position in recombination step of four-gamete test method and ARG4WG algorithm that directly affect to the number of recombination events in ARG building process are pointed out in section III. The GAMARG algorithm will also be proposed in this section. The performance of our algorithm in comparison to other heuristics and exhaustive search methods is discussed in section IV. Finally, we conclude our work and suggest some future works.

## 2. ARG BUILDING PROBLEM

Given a set D = $\{S_1, \ldots, S_N\}$ of $N$ input sequences (haplotypes), $S_x$ has $m$ markers, $1 \le x \le N$; $S_x[i]$ denotes site $i$ of $S_x$ that has the value of either 0 (one of the SNP alleles), or 1 (another allele), $1 \le i \le m$. The ARG building problem is to construct the relationships between sequences in D through three events: coalescence, mutation, and recombination. The coalescence events merge two identical sequences to one. The mutation event makes change in a site of the sequence. The recombination event breaks one sequence to two subsequences, one subsequence contains the prefix of the sequence and one contains the suffix of the sequence. Our task is to build an ARG with the minimum number of recombination events under the infinite-sites assumption.

Consider a data set D(5) = $\{S_1, S_2, S_3, S_4, S_5\}$ of 5 sequences with 5 sites as below:

| | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| $S_1$ | = | 1 | 1 | 0 | 0 | 1 |
| $S_2$ | = | 0 | 1 | 0 | 1 | 0 |
| $S_3$ | = | 0 | 0 | 0 | 1 | 0 |
| $S_4$ | = | 1 | 0 | 0 | 0 | 0 |
| $S_5$ | = | 1 | 0 | 0 | 0 | 1 |

Figure 1 is an example of an ARG building process for D(5). An ARG with 12 events (numbered from 1 to 12 in circles) is built backward in time, starting from the input data, until a single common ancestor (10001) is reached. A "*" at a site denotes a non-ancestral material. 3 different evolutionary events are performed in the building process. A recombination event as in the state 1 breaks a sequence 01010 into two sub-sequences: 01*** contains the prefix and **010 contains the suffix of the sequence 01010. A coalescence event as in the state 2 combines two sequences **010 and 00010 into one sequence 00010. A mutation event as in the state 3 changes a mutated site 4 from 1 to 0.

## 3. METHOD

### 3.1 Four-Gamete Test

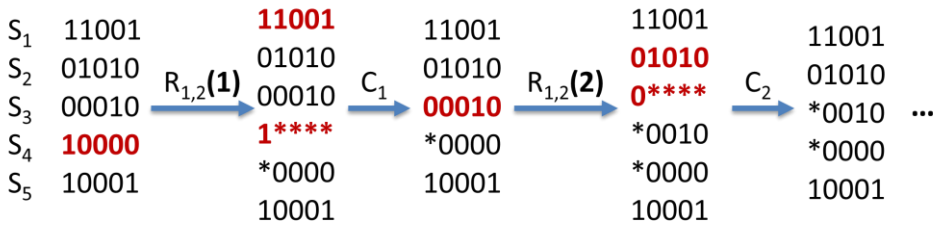
Under the infinite-site assumption, we called two sites $i$ and $j$ incompatible if they contain all four gametic types 00, 01, 10, 11 [3]. There will be at least one recombination event between two incompatible sites $i$ and $j$. The exhaustive methods aim to find out the optimal breakpoints, that is, the smallest number of recombination events, to break all these incompatible sites.

Let $FreqGamete_{i,j} = \{freq00_{i,j}, freq01_{i,j}, freq10_{i,j}, freq11_{i,j}\}$ be the frequencies of gametic types 00, 01, 10, 11 occurring between sites $i$ and site $j$, respectively.

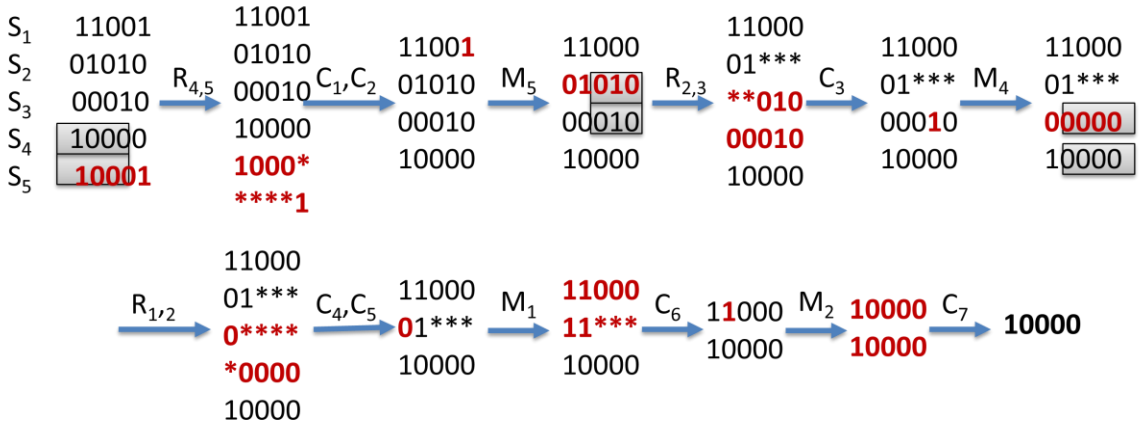
In the data set D(5), there are three pairs of incompatible sites: site 1 and site 2; site 2 and site 4; site 2 and site 5. The frequencies of 4 gametic types are $FreqGamete_{1,2} = \{1,1,2,1\}$; $FreqGamete_{2,4} = \{2,1,1,1\}$; $FreqGamete_{2,5} = \{2,1,1,1\}$, respectively. In this case, at least two recombination events must be happened in the evolutionary history of the sequences. Figure 1 is a minimal ARG with two recombination events representing the evolutionary history of data set D(5).

We observed that there are three gametic types having frequency 1. So breaking one of these gametic types between these sites will give us better solution. For example, performing a recombination between site 1 and site 2 on one of three sequences $S_1$, $S_2$, $S_3$ will break this pair of incompatible sites. However, as $freq10_{1,2}$ equals 2, so if we perform a recombination between site 1 and site 2 on one of two sequences $S_4$, $S_5$, we just reduce the frequency of occurrence of gametic type 10 by one and we do not break this pair of incompatible sites. In this case, we need one more recombination event to break this pair of incompatible sites (Figure 2a).

### 3.2 ARG4WG Algorithm




**(a)**




**(b)**

**Figure 2. ARG building process for data set D={ $S_1$, $S_2$, $S_3$, $S_4$, $S_5$} (a) based on four gametic tests and (b) in ARG4WG algorithm** $\xrightarrow{R_{i,j}}$ denotes a recombination event between site i and site j; $\xrightarrow{C_x}$ denotes *x*th coalescence event; $\xrightarrow{M_i}$ denotes a mutation event at site i. (a) The ARG building process started by choosing $S_4$ to do a recombination event between site 1 and site 2 ($R_{1,2}$**(1)**). As $freq01_{1,2} = 2$, this recombination event help to reduce the frequency of gametic type 01 between site 1 and site 2 by one and $FreqGamete_{1,2} = \{1,1,1,1\}$ on the next generation. So we need to do one more recombination event between those sites ($R_{1,2}$**(2)**) to break this pair of incompatible sites. So this choice (and also the same with $S_5$) will waste two recombination events while choosing $S_1$, $S_2$, $S_3$ (that all have the frequency of occurence of gametic type equal 1) to break between those sites just waste one recombination events. (b) The longest shared end is detected between $S_4$ and $S_5$ (covered by rectangles), a recombination event between site 4 and site 5 is putted on $S_4$ (or $S_5$) to produce 2 subsequences. By this way, ARG4WG always need 3 recombination events to build ARGs for this data set.

Working backward in time, ARG4WG first performs all possible coalescence and mutation events. The algorithm then searches for a pair of sequences that have the longest shared ends, that is, the longest match in term of ancestral material from the left or the right of two sequences. A recombination is performed on a sequence to break a sequence into two subsequences. A subsequence containing the longest shared region will be coalesced with the remaining sequence right after the recombination step.

The longest shared ends strategy helps ARG4WG to work with thousands of whole genome sequences. It aims to build plausible ARGs and cannot give us the minimal ARGs. Figure 2b illustrates briefly the way ARG4WG works with data set D(5). As we see, in this case, ARG4WG always performs recombination on $S_4$ or $S_5$ first. This choice does not give us the optimal solution and require at least 3 recombination events to build an ARG.

## 3.3 GAMARG Algorithm

We propose GAMARG algorithm that combines the four-gamete test constraint with the longest shared ends strategy in ARG4WG to optimize the number of recombination events in ARG building process.

As using four-gamete test to build minimal ARG is not possible for large datasets. From the observation described in Section 3.1, we propose a simplification of the four-gamete test by considering only pairs of incompatible sites having frequency 1 for at least one gametic type. This assumption guarantees that we always break at least one pair of incompatible sites when performing a recombination between a pair of incompatible sites *i* and *j*.

Let $\delta$ be a size of sliding window that we will scan to find all pairs of incompatible sites in this region. In particular, we scan through all markers. For each marker *i* ($0 \le i < m$), we will scan to find all pairs of incompatible sites in a range [*i*, *i*+ $\delta$].

Let $S_x(i,j)$ be a sequence containing a gametic type with frequency 1 at a pair of incompatible sites $i$ and $j$ ($0 \le i < m, j - i \le \delta$). That is, $S_x(i,j)$ satisfies the following conditions:

$$\begin{cases} freq00_{i,j} > 0 \; and \; freq01_{i,j} > 0 \; and \; freq10_{i,j} > 0 \; and \; freq11_{i,j} > 0 \\ freq00_{i,j} = 1 \; or \; freq01_{i,j} = 1 \; or \; freq10_{i,j} = 1 \; or \; freq11_{i,j} = 1 \end{cases}$$

We use the same definitions as in [1]:

- $S_x[i]$ matches $S_y[i]$ if $S_x[i] = S_y[i]$ or $S_x[i] = *$ or $S_y[i] = *$.
- $(S_x,S_y)\{d,l\}$ is a shared end pair of sequence $S_x$ and sequence $S_y$ with the maximal matching length $l$ from the left ($d = left$) or from the right ($d = right$).
- $(S_x,S_y)\{d,l\}$ exists if and only if there are at least one marker $i$ in matching region that $S_x[i] = S_y[i] \neq *$.

For a shared end pair $(S_x,S_y)\{d,l\}$, following the longest shared end strategy, the breakpoint is specified between:

- $l$ and $l + 1$ where $d = left$ and $S_x[i]$ match $S_y[i]$ for all $1 \le i \le l$ and $S_x[l+1] \neq S_y[l+1]$.
- $l$ -1 and $l$ where $d = right$ and $S_x[i]$ match $S_y[i]$ for all $l \le i \le m$ and $S_x[l-1] \neq S_y[l-1]$.

Given a candidate sequence $S_x(i,j)$, we need to find the best breakpoint in range $[i,j]$. We once again tackle this problem by using the longest shared end strategy. We find out the longest shared end between this sequence and all other sequences. If there exists a sequence $S_z$ that a shared end pair $(S_x, S_z)\{d,l\}$ satisfies $i \le l \le j$, then $S_x$ will be broken at marker $l$ as mentioned above. If no shared end pair in range $[i,j]$ exists, the breakpoint is chosen randomly between site $i$ and $i$+1 or between site $j$-1 and $j$.

**GAMARG algorithm:** The GAMARG algorithm starts from time $t = 1$. The set of sequences at time $t$ is denoted as $D_t$ ($D_1$=D). For each $D_t$, the candidate lists for coalescence, mutation and recombination events are constructed as the following:

- Coalescence list **C**: For a shared end pair $(S_x,S_y)\{d,l\}$ of sequences $S_x$ and $S_y$, if $l = m$, then $(S_x,S_y)\{d,l\}$ is added into the coalescence list.
- Mutation list **M**: For a marker $i$ ($1 \le i \le m$), if $S_x[i] = 1$ and $\forall S_y \in D_t \setminus \{S_x\}: S_y[i] \neq 1$ or $S_x[i] = 0$ and $\forall S_y \in D_t \setminus \{S_x\}: S_y[i] \neq 0$, then $S_x[i]$ is added into mutation list.
- Gamete list **G**: For a pair of incompatible sites $(i,j)$ ($0 \le i < m$, $j - i \le \delta$), if exist a sequence $S_x$ that contains a gametic type with frequency 1, then $S_x(i,j)$ is added into gamete list.
- Shared-end list **S**: For a shared end pair $(S_x,S_y)\{d,l\}$ of sequences $S_x$ and $S_y$, if $0 < l < m$, $(S_x,S_y)\{d,l\}$ is added into the recombination list.

When one of three events occurs, the next sequence set $D_{t+1}$ is created from the current sequence set $D_t$ as described below and four candidate lists are updated.

- If a coalescent event occurs between two sequences $S_x$ and $S_y$, two sequences $S_x$ and $S_y$ are merged into a common ancestor $S'$:
$$D_{t+1} = \left(D_t \backslash \{S_x, S_y\}\right) \cup \{S'\}.$$
- If a mutation event occurs on a sequence $S$, a new sequence $S'$ is created from sequence $S$ with the mutation:
$$D_{t+1} = (D_t \backslash \{S\}) \cup \{S'\}.$$
- If a recombination occurs on a sequence $S_x(i,j)$, a breakpoint is put in [i,j]. Two new subsequences $S_{x1}$ and $S_{x2}$ are created from sequence $S_x$:
$$D_{t+1} = (D_t \backslash \{S_x\}) \cup \{S_{x1}, S_{x2}\}$$
- If a recombination occurs on a shared end pair $(S_x, S_y)\{d,l\}$, pick a sequence having less ancestral material in its shared end part to do recombination. Assuming $S_x$ is chosen, sequence $S_x$ will be broken into two new subsequences $S_{x1}$ and $S_{x2}$:
$$D_{t+1} = (D_t \backslash \{S_x\}) \cup \{S_{x1}, S_{x2}\}$$

**The GAMARG algorithm**

**Input:** A set of $N$ sequences with $m$ markers (snps)

**Output:** An ARG containing coalescence, mutation and recombination events among sequences**.**

- **Step 1:** If Coalescence list **C** is not empty, do all possible coalescence events.
- **Step 2:** If Mutation list **M** is not empty, do all possible mutation events then go to Step 1. If no mutation possible, go to Step 3.
- **Step 3:** If Gamete list **G** is not empty, do a recombination then go to Step 1.
- **Step 4:** If Shared-end list **S** is not empty, do a recombination followed by a coalescence. Go to Step 1.
- **Step 5:** Repeat Step 1, Step 2 and Step 3, Step 4 until a single common ancestor is reached.

Candidates from four lists are selected as the following:

- The candidate from the coalescence list or the mutation list to perform coalescence or mutation is taken randomly.
- In the Gamete list, if a candidate sequence $S_x(i, j)$ having the shortest distance from site $i$ to site $j$, that is, $(j - i)$ has the smallest value, $S_x$ is the first priority to perform recombination. If there is more than one candidate having the same shortest distance, we will choose one randomly.
- In the Shared-end list, the pair of sequences with the longest shared end in term of ancestral material will be the first choice for recombination. If there is more than one candidate having the same longest shared end, one is picked randomly.

The random choices in GAMARG algorithm result in different ARGs for different runs.

# 4. EXPERIMENTS AND RESULTS

To evaluate the performance of GAMARG, we conducted experiments on different datasets. First, we measured GAMARG, Margarita, ARG4WG, REARG, and exhaustive algorithms on Kreitman's dataset [9] that included 11 sequences of length 43. This small dataset is a benchmark used in evaluating the performance of many algorithms either to find lower bound of recombination or to build minimal ARGs.

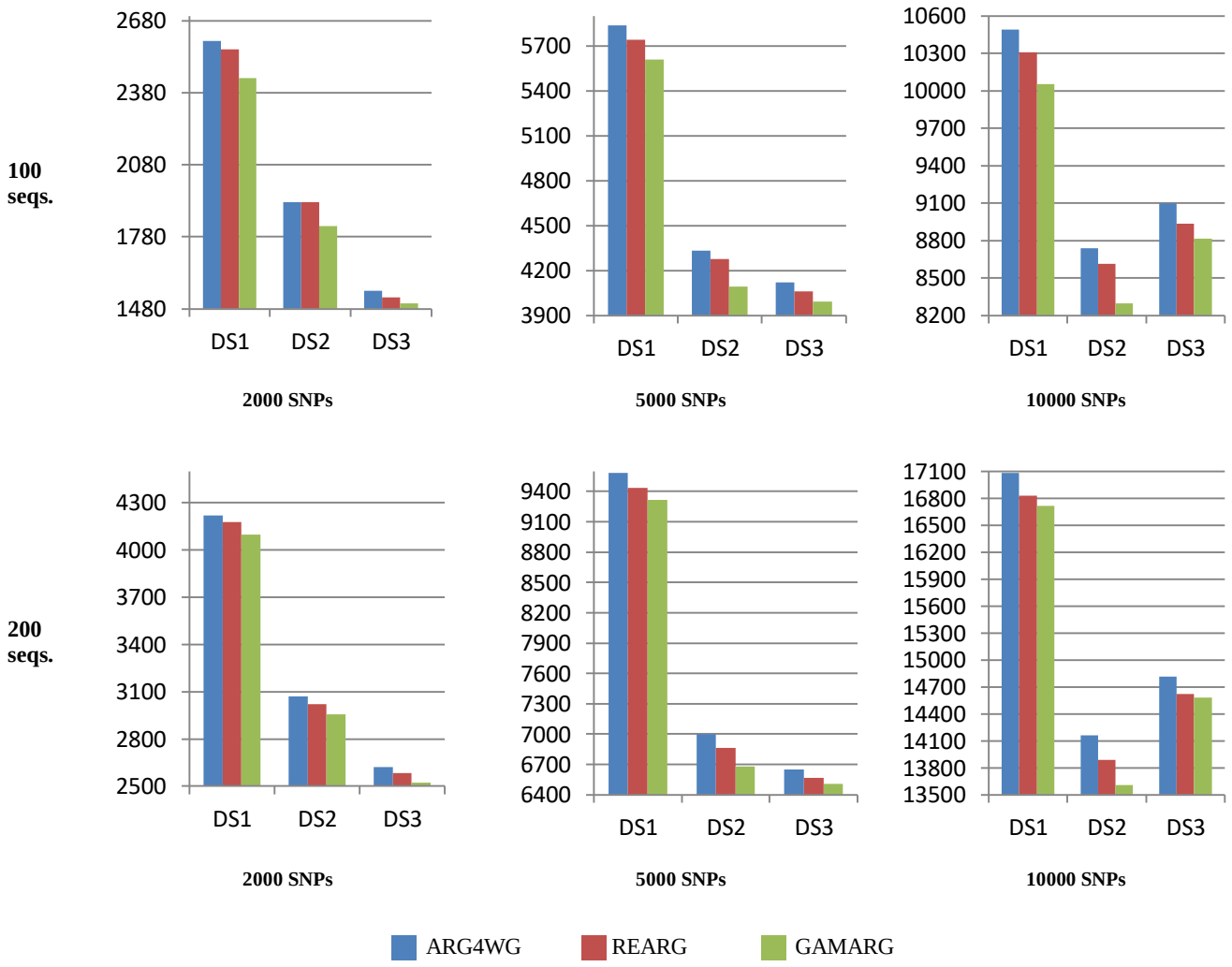



**Figure 3. The smallest number of recombination events found by 3 algorithms for 100 and 200 haplotypes with 2000, 5000, and 10000 SNPs of DS1, DS2, and DS3**

Second, we tested all 4 above algorithms on two simulation datasets: SDS1 included 50 sequences of length 54 and SDS2 included 75 sequences of length 45 that were public at https://people.eecs.berkeley.edu/~yss/lu.html.

Third, we examined GAMARG algorithm on the datasets used in [7] that extracted from the 1000 Genomes Project [10]. Note that experiment results from [7] showed that Margarita was not stable and needed a huge number of recombination events to build an ARG for these datasets. We compared GAMARG with ARG4WG and REARG in terms of the number of recombination events and the runtime. We could not perform exhaustive search methods as they were not applicable for these large datasets.

REARG has three versions called REARG_SIM, REARG_LEN, REARG_COM. The output of REARG is the best output from all these versions.

## 4.1 Kreitman's Dataset

1000 ARGs were built by each algorithm and we recorded the ARG having the smallest number of recombination events. ARG4WG and REARG got ARG with **10** recombination events as their best results. Margarita could build an ARG with **8** recombination events. The GAMARG could generate different ARGs with **7** recombination events using $18 \leq \delta < 43$. This result is the optimal solution as is also found by exhaustive search methods [3], [4]. This result shows that GAMARG is as good as exhaustive searches for small datasets. Moreover, it takes only 8 seconds to build 1000 ARGs (i.e., as fast as ARG4WG).

## 4.2 Simulation Datasets

10000 ARGs were built by each algorithm on each dataset and we recorded the ARG having the smallest number of recombination events. We ran GAMARG with different $\delta$ and we had best results

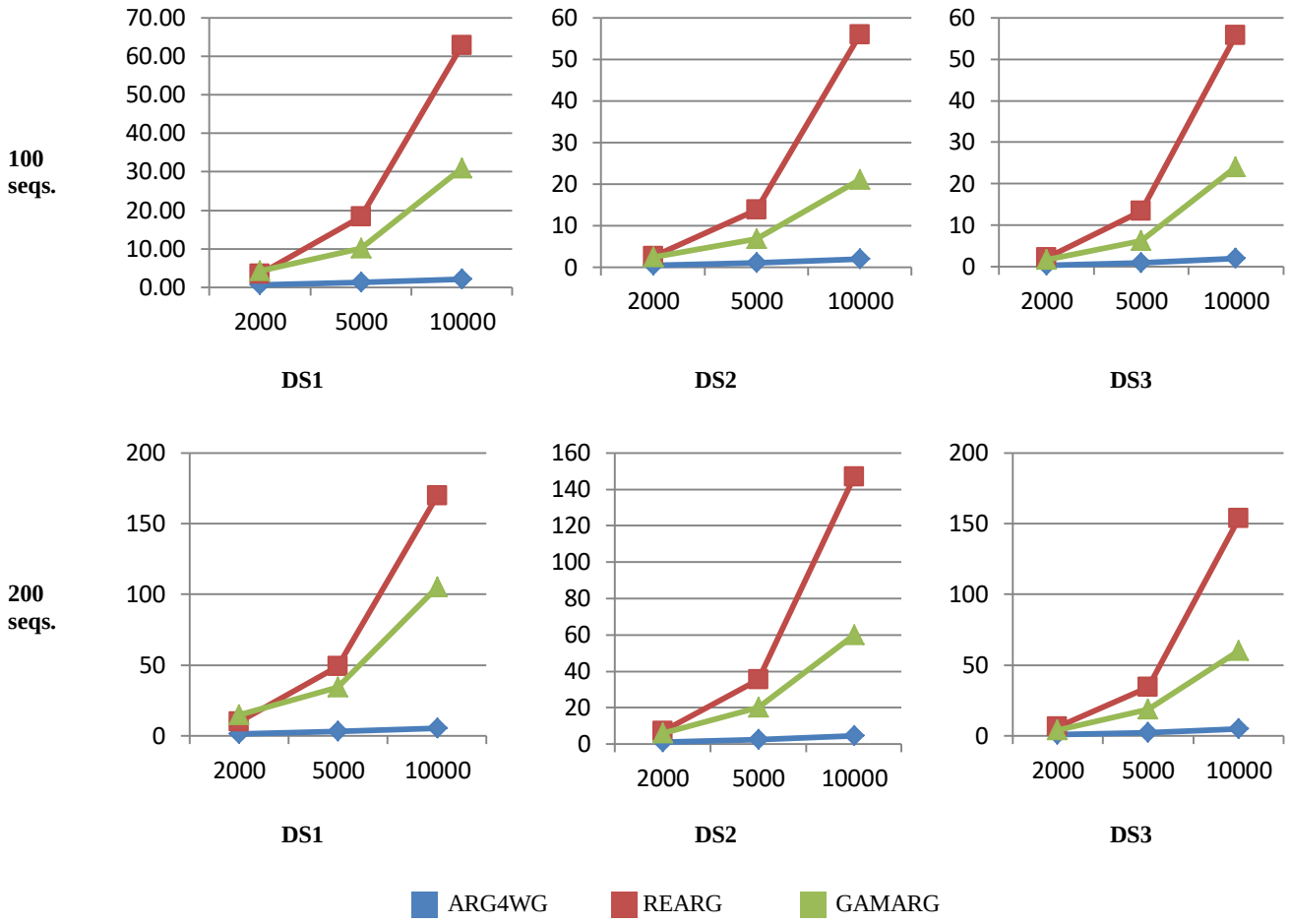



**Figure 4. Average of runtimes (second) of ARG4WG, REARG, and GAMARG for 100 and 200 haplotypes with 2000, 5000, and 10000 SNPs of DS1, DS2, and DS3 datasets**

with $29 \leq \delta < 54$ for SDS1 and $11 \leq \delta < 45$ for SDS2. The results of all algorithms are described in Table 1.

The experiment results show that GAMARG can reach to minimal ARGs for SDS1 and only one recombination more than the optimal solutions for SDS2. The results of Margarita, ARG4WG, and REARG are very far from the optimal solutions.

**Table 1. The results from different algorithms on simulated datasets**

| | **SDS1** | **SDS2** |
|---|---|---|
| **Minimal ARG** | **10** | **12** |
| **Margarita** | 14 | 18 |
| **ARG4WG** | 17 | 18 |
| **REARG** | 17 | 20 |
| **GAMARG** | **10** | **13** |

## 4.3 Datasets from the 1000 Genomes Project

We compared the runtime and the number of recombination events on 18 datasets of 100, 200 haplotypes with 2000, 5000, 10000 SNPs extracted from 3 different regions (i.e. DS1, DS2, DS3) of Chromosome 1 from the 1000 Genomes Project.

As in [7], on each data set, 1000 ARGs were built by each algorithm and the ARG with the smallest number of recombination events was recorded. In these tests, we ran GAMARG using $\delta = 5$.

Experiment results (see Figure 3) show that GAMARG algorithm produces ARGs with much smaller number of recombination events in comparison to that of ARG4WG and REARG in all tests. The outperformance of GAMARG in comparison to other algorithms is clearly significant for 100 sequences. For larger datasets with more sequences, the diversity of the data increases. Thus, there are many incompatible sites, however, only few of many of them might satisfy the constraint that at least one gametic type having frequency 1. In this case, the advantage of GAMARG over ARG4WG and REARG is not very significant.

The average running times to build an ARG by each algorithm were calculated for each test. As shown in Figure

4, for 2000 SNPs, there are not much different in the running times between algorithms. For longer sequences (i.e., 5000 and 10000 SNPs), GAMARG is slower than ARG4WG but faster than REARG.

## 4.4 Discussion

Four-gamete test is the well-known technique in computing the minimal recombination ARG for small datasets. The longest shared end strategy in ARG4WG algorithm is very effective for large datasets. The combination of them in GAMARG algorithm allows it not only to work with thousands sequences with tens of thousands of SNP markers but also to find minimal recombination ARGs.

The results on small datasets indicate that both ARG4WG and REARG algorithms are not suitable for small datasets. The longest shared segment strategy of Margarita has obtained the better results than ARG4WG and REARG for small datasets. However, this strategy causes Margarita much more recombination events and runtime than ARG4WG and REARG for medium or large datasets [6], [7].

The proposed GAMARG algorithm performs well in all cases, not only for small datasets but also for large datasets. However, we need to investigate the best choice for $\delta$ parameter more. For small datasets, it is not a problem because GAMARG requires only small time to build thousands ARGs.

For human genome data set, we examined GAMARG with different values for $\delta$ (i.e., 5, 10, 15, 20, 25, and 30) on different datasets with different sizes. 5000 ARGs were built and ARG with the smallest number of recombination events was recorded on each dataset. The results show that GAMARG produces similar results while $\delta$ has one of values 5, 10, 15 for 500 SNPs. However, for longer sequences (i.e., 1000 and 2000 SNPs), the algorithm works best in term of number of recombination events with $\delta = 5$.

# 5. CONCLUSION

Constructing minimal ARGs from large datasets is still an open problem. ARG4WG algorithm can build ARG for thousands of whole genome sequences, however, it is not designed to construct minimal ARGs. In this work, we propose GAMARG algorithm that combines four-gamete test with the longest shared end strategy in recombination step to optimize the number of recombination events in ARG building process. The GAMARG algorithm infers ARGs with smaller number of recombination events than all other heuristic methods. Specially, the GAMARG algorithm can competitive with exhaustive search methods as it can find minimal ARGs for small datasets in very little time.

In the future, we will consider more about methods to calculate the haplotype blocks to have a better estimation for parameter $\delta$.

# 6. ACKNOWLEDGMENTS

We thank Centre for Informatics Computing (VAST) for allowing us to use their HPC. This research is supported by Vietnam Academy of Science and Technology (ĐLTE00.01/19-20).

# 7. REFERENCES

[1] M. Arenas, “The importance and application of the ancestral recombination graph,” *Front. Genet.*, vol. 4, p. 206, 2013.

[2] L. Wang, K. Zhang, and L. Zhang, “Perfect phylogenetic networks with recombination,” *J. Comput. Biol.*, vol. 8, no. 1, pp. 69–78, 2001.

[3] Y. S. Song and J. Hein, “Constructing minimal ancestral recombination graphs,” *J. Comput. Biol.*, vol. 12, no. 2, pp. 147–169, 2005.

[4] R. B. Lyngsø, Y. S. Song, and J. Hein, “Minimum recombination histories by branch and bound,” in *International Workshop on Algorithms in Bioinformatics*, 2005, pp. 239–250.

[5] M. J. Minichiello and R. Durbin, “Mapping trait loci by use of inferred ancestral recombination graphs,” *Am. J. Hum. Genet.*, vol. 79, no. 5, pp. 910–922, 2006.

[6] T. T. P. Nguyen, V. S. Le, H. B. Ho, and Q. S. Le, “Building ancestral recombination graphs for whole genomes,” *IEEE/ACM Trans. Comput. Biol. Bioinforma.*, vol. 14, no. 2, pp. 478–483, 2017.

[7] T. T. P. Nguyen and V. S. Le, “Building minimum recombination ancestral recombination graphs for whole genomes,” in *2017 4th NAFOSTED Conference on Information and Computer Science, NICS 2017 - Proceedings*, 2017, vol. 2017–Janua, pp. 248–253.

[8] R. R. Hudson and N. L. Kaplan, “Statistical properties of the number of recombination events in the history of a sample of DNA sequences,” *Genetics*, vol. 111, no. 1, pp. 147–164, 1985.

[9] M. Kreitman, “Nucleotide polymorphism at the alcohol dehydrogenase locus of Drosophila melanogaster,” *Nature*, vol. 304, no. 5925, p. 412, 1983.

[10] 1000 Genomes Project Consortium and others, “A map of human genome variation from population-scale sequencing,” *Nature*, vol. 467, no. 7319, p. 1061, 2010.