Bonnet *et al. Algorithms for Molecular Biology (2023) 18:15*
https://doi.org/10.1186/s13015-023-00241-3

Algorithms for Molecular Biology

RESEARCH

Open Access

Constructing founder sets under allelic and non-allelic homologous recombination

Konstantinn Bonnet[1], Tobias Marschall[1*] and Daniel Doerr[1*]

Abstract

Homologous recombination between the maternal and paternal copies of a chromosome is a key mechanism for human inheritance and shapes population genetic properties of our species. However, a similar mechanism can also act between different copies of the same sequence, then called *non-allelic homologous recombination (NAHR)*. This process can result in genomic rearrangements—including deletion, duplication, and inversion—and is underlying many genomic disorders. Despite its importance for genome evolution and disease, there is a lack of computational models to study genomic loci prone to NAHR. In this work, we propose such a computational model, providing a unified framework for both (allelic) homologous recombination and NAHR. Our model represents a set of genomes as a graph, where haplotypes correspond to walks through this graph. We formulate two founder set problems under our recombination model, provide flow-based algorithms for their solution, describe exact methods to characterize the number of recombinations, and demonstrate scalability to problem instances arising in practice.

Keywords Founder set reconstruction, Variation graph, Pangenomics, NAHR, Homologous recombination

Background

Twenty years ago, Esko Ukkonen introduced the problem of inferring founder sets from haplotyped single nucleotide polymorphism (SNP) sequences under allelic recombination [1]. Ukkonen's work has since inspired a wealth of research addressing various aspects and applications of founder set reconstruction ranging from the reconstruction of ancestral recombinations and pangenomics to applications in phage evolution [2–4]. In its original setting, the problem sets out from a given set of m sequences of equal length n, where characters across sequences residing at the same index position correspond to a SNP. It then asks for a smallest set of sequences, called *founder set*, such that each given sequence can be constructed through a series of crossovers between sequences of the founder set, where each segment between two successive recombinations must meet a minimum length threshold. The *Minimum Founder Set* problem is NP-complete in general [5], but is solvable in linear time for the special case of founder sets of size two [1, 6]. Since its introduction, various heuristics and approximations have been proposed [6–8]. A variant of this problem restricts crossovers to coincide at certain positions, thereby decomposing the input sequences into a universally shared succession of blocks. The resulting problem, known as *Minimum Segmentation Problem* is polynomial [9]. In his seminal paper, Ukkonen devised a $O(n^2m)$ algorithm for its solution which has been substantially improved by Norri et al. [10] to linear time, i.e. $O(nm)$ by exploiting the positional Burrows-Wheeler transform [11].

Just like the Minimum Founder Set Problem, the vast majority of population genetic analyses and genome-wide

Tobias Marschall and Daniel Doerr are Joint last authors.

*Correspondence:
Tobias Marschall
tobias.marschall@hhu.de
Daniel Doerr
daniel.doerr@hhu.de
[1] Institute for Medical Biometry and Bioinformatics, Medical Faculty, and Center for Digital Medicine, Heinrich Heine University, Moorenstr. 5, 40225 Düsseldorf, Germany

association studies have been focused on SNPs in the past decades, neglecting more complex forms of variation—mostly for technical difficulties in detecting them. In particular, structural variants (SVs), commonly defined as variants of at least 50bp, have posed substantial challenges and studies based on short sequencing reads typically detect less than half of all SVs present in a genome [12]. Recent technological and algorithmic advances help to overcome these limitations [13]. Long read technologies now enable haplotype-resolved *de novo* assembly of human genomes [14], which in turn enables a much more complete ascertainment of SVs [15]. In 2022, the first complete telomere-to-telomere assembly of a human genome was announced [16], heralding a new era of genomics where high-quality, haplotype-resolved assemblies of complex repetitive genomic structures become broadly available. Presently, the Human Pangenome Reference Consortium (HPRC), is applying these techniques to generate a large panel of haplotype-resolved genome assemblies from samples of diverse ancestries [17, 18]. These emerging data sets enable studying genetic loci involving duplicated sequence, called *segmental duplications (SDs)*, which are amenable to NAHR, are therefore highly mutable, and show complicated evolutionary trajectories [19, 20]. The T2T-CHM13 study alone reports over 40 thousand segmental duplications that amount to 202Mb (6.6% of the human genome) [16].

Interestingly, at loci with highly similar segments arranged in opposite orientations, such as Segment 3 in Fig. 1, NAHR can lead to *inversion*, i.e. the reversal of the interior sequence (Segment 4 in Fig. 1). Because of being flanked by a pair of copies of the same sequence (cf. Segment 3) that often comprises tens of thousands of bases, such events have been largely undetectable by sequencing technologies with read lengths below the length of the duplicated sequence; in particular by conventional short read sequencing. Recent studies applying multiple technologies reveal that inversions affect tens of megabases

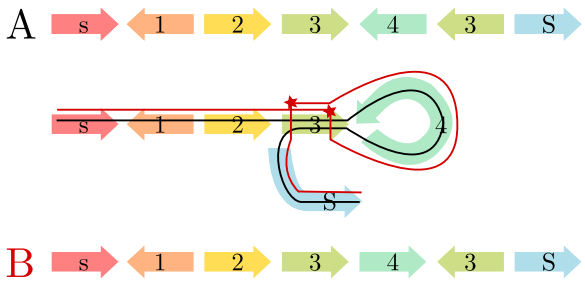


Fig. 1 *Illustration of an NAHR-mediated inversion.* Haplotype **A** (black line) represents the original configuration, while haplotype **B** (red line) can be derived from *A* by two recombination events between inverted repeats of genomic marker 3 as indicated by the red stars

of sequence in a typical human genome [21]. Unlike most other classes of genetic variation, inversions are often *recurrent* with high mutation rates, that is, the same events have happened multiple times in human history [22]. Depending on the structures of duplicated sequence at a particular locus, individual human haplotypes can differ in their potential for NAHR. This can have important implications for the risk for a range of genetic disorders caused by NAHR-mediated mutations [22].

In the past two decades, various mathematical models and algorithms to study genome rearrangements have been proposed. These range from the classic reversal [23, 24] and transposition [25] model to composed models for two or more balanced rearrangements [26, 27], to generalized models such as the popular *Double Cut and Join (DCJ)* [28, 29]. As the research in this field continues, advanced models can additionally accommodate one or more types of unbalanced rearrangements, i.e., deletion, insertion, and duplication [30, 31]. Yet, none of these models adequately considers sequence similarity as a prerequisite for NAHR, which is a key molecular mechanism shaping many complex loci in the human genome. In summary, there are now technological opportunities to study the population history of recalcitrant SD loci that are prone to genome rearrangements and relevant to disease, but computational models to facilitate this have so far been lacking.

This work addresses this deficit by proposing a rearrangement model that is based on the molecular mechanism of homologous recombination and by solving variants of Ukkonen's Founder Set Problem that can provide insights into the evolution of complex loci driven by NAHR. The genome model underlying the approach at hand represents DNA sequences at a level of abstraction where they are already decomposed into genomic markers with assigned homologies. Here, our notion of homology is a synonym for *high DNA sequence similarity*, as we adopt the terminology underlying the concept of homologous recombination. Our model permits recombination events to occur between homologous markers independent of their position within or between haplotypes, as long as the markers' orientations are respected. In other words, a marker can only recombine with a homologous marker alongside the same direction, as illustrated by Fig. 1, because a recombination event can only occur between homologous markers if they are aligned to each other. By virtue of recapitulating the underlying NAHR, it implicitly allows for all the rearrangements this molecular mechanism can give rise to, including deletion, duplication, and inversion.

Marker decomposition and homology assignment can be done in practice with genome graph building tools such as MBG [32], minigraph [33], or pggb [34]. Our algorithms can work with any *variation graph* or

pangenome graph with nodes corresponding to homologous DNA segments and edges between segments corresponding to observed adjacencies in a given set of haplotypes.

Methods

Preliminaries

A *(genomic) marker m* is an element of the finite universe of markers denoted by $\mathcal{M}$, and is associated with a fragment of a double-stranded DNA molecule. Each marker can be traversed in *forward* and *reverse* direction. A marker in forward orientation (which is the default orientation) is traversed from left to right. Overline notation $\overline{m}$ indicates the reversal of a marker m, which is carried out relative to its orientation, i.e., $\overline{\overline{m}} = m$. Similarly, $\overline{\mathcal{M}}$ represents the set of all reversed markers. We designate two forward markers $\{s, S\} \subseteq \mathcal{M}$ as *terminal markers*. In what follows, we study *terminal sequences*, that is, sequences drawn from the alphabet of oriented markers $\mathcal{M} \cup \overline{\mathcal{M}}$ that start with s or $\overline{S}$, end in S or $\overline{s}$ and do not contain any further terminal markers in between. A terminal sequence can be traversed in forward and reverse direction. A *haplotype* is a terminal sequence that starts with s (*source*) and ends with S (*sink*).

Example 1 Consider in the following two sequences of genomic markers A and X drawn from the universe of markers $\mathcal{M} = \{\mathtt{s}, \mathtt{1}, \mathtt{2}, \mathtt{3}, \mathtt{4}, \mathtt{S}\}$, where $A = \mathtt{s}\overline{\mathtt{1}}\mathtt{23}\overline{\mathtt{43}}\mathtt{S}$ and $X = \mathtt{s}\overline{\mathtt{1}}\mathtt{234}\overline{\mathtt{321}}\overline{\mathtt{s}}$. Sequence A starts and ends with terminal markers $\mathtt{s}$ and $\mathtt{S}$, respectively, thus constituting a *haplotype* over $\mathcal{M}$. Conversely, X starts with $\mathtt{s}$ and ends in $\overline{\mathtt{s}}$ and therefore is a terminal sequence, but not a *haplotype*.

Given a sequence A, $|A|$ indicates the length of A which corresponds to the number of A's constituting elements. $\overline{A}$ defines the *reverse complementation* of sequence A, i.e., the simultaneous reversal of the sequence and its constituting elements. The element at the ith position in sequence A is denoted by $A[i]$. A *segment* of sequence A starting at position i and ending at and including position j is denoted by $A[i..j]$. Then, $A[..i] := A[1..i]$ and $A[i..] := A[i..|A|]$ denote the *prefix* and *suffix* of A, respectively. Given two sequences A and B, then $B \lhd A$ indicates that B is a segment of A, i.e., $|B| \leq |A|$ and there exists some $i \in [1, |A| - |B|]$ with $B = A[i..i + |B| - 1]$. Finally, the operator "+" indicates the concatenation of two sequences.

Example 1 (cont'd) The length of A is $|A| = 7$; its reverse complement is $\overline{A} = \overline{\mathtt{S}}\mathtt{34}\overline{\mathtt{32}}\mathtt{1}\overline{\mathtt{s}}$; $A[4..6]$ is a segment of A and corresponds to sequence $\mathtt{3}\overline{\mathtt{43}}$, and consequently $\mathtt{3}\overline{\mathtt{43}} \subseteq A$ holds true; The segments $X[..6] = \mathtt{s}\overline{\mathtt{1}}\mathtt{23}\overline{\mathtt{4}}$ and $A[7..] = \mathtt{S}$ are a prefix and a suffix of X and A, respectively; The concatenation of prefix $X[..6]$ and suffix $A[7..]$ results in haplotype $X[..6] + A[7..] = \mathtt{s}\overline{\mathtt{1}}\mathtt{23}\overline{\mathtt{4}}\mathtt{S}$.

A *recombination* is an operation that acts on a shared oriented marker m of any two terminal sequences A and B: let $A[i] = B[j] = m$; recombination $\chi(A, B, i, j)$ produces terminal sequence $C = A[..i] + B[j + 1..]$. For a given set of haplotypes $\mathcal{H}$, span $(\mathcal{H})$ denotes the *span*, i.e., the set of all *haplotypes* generated by applying χ on haplotypes $\mathcal{H}$ and the resulting terminal sequences. More precisely, let Θ be the universe of terminal sequences, defined recursively by $\mathcal{H} \cup \overline{\mathcal{H}} \subseteq \Theta$ such that for any $A, B \in \Theta$ with some $A[i] = B[j]$ the recombinant $C = A[..i] + B[j + 1..]$ and its reverse complement $\overline{C}$ is also in Θ. Then span $(\mathcal{H}) := \{A \in \Theta \mid A \text{ is a haplotype}\}$. Accordingly, we also say that "$\mathcal{H}$ is a *generating set* of span $(\mathcal{H})$". Conversely, given any (possibly infinite) set of haplotypes $\mathcal{S}$ and some $\mathcal{H} \subseteq \mathcal{S}$, $\mathcal{H}$ is a generating set of $\mathcal{S}$ if and only if span $(\mathcal{H}) = \mathcal{S}$.

Example 1 (cont'd) Recombination $\chi(A, \overline{A}, 4, 2)$ produces terminal sequence $X = \mathtt{s}\overline{\mathtt{1}}\mathtt{234}\overline{\mathtt{321}}\overline{\mathtt{s}}$. Subsequent recombination $\chi(X, A, 6, 6)$ produces haplotype $B = \mathtt{s}\overline{\mathtt{1}}\mathtt{23}\overline{\mathtt{4}}\mathtt{3S}$. If $\{A\}$ is a given set of haplotypes, then span $(\{A\}) = \{A, B\}$.

In this paper, we study the following three problems:

Problem 1 (Founder Set) Given a set of haplotypes $\mathcal{H}$, find a generating set $\mathcal{F}$ such that span $(\mathcal{F})$ = span $(\mathcal{H})$ and $\sum_{A \in \mathcal{F}} |A|$ is minimized.

We minimize total length because current knowledge on the evolution of complex genomic loci indicates their contained segmental duplications often causes them to expand over time [35]. Consequently we expect ancestral loci to be more compact and contain fewer duplications. We prefer this formulation over minimizing the founder set's cardinality, because the latter would allow for solutions with founder sequences of unbounded length, which is biologically irrelevant. We call a solution to Problem 1 a *founder set* and its members *founder sequences*. The following problem is related to Ukkonen's Minimum Segmentation Problem [1]:

Problem 2 (Recombination Count) Given a set of terminal sequences $\mathcal{T}$ and a terminal sequence Q,

count the number of recombinations r of the form $A_{k+1} = \chi(A_k, T_k, \cdot, \cdot)$, with $0 \leq k \leq r$ and $T_0, .., T_r \in \mathcal{T}$, that are necessary to generate Q from $\mathcal{T}$, i.e., $A_0 \in \mathcal{T}$ and $Q = A_r$, if feasible and report its infeasibility otherwise.

At last, the combination of Problems 1 and 2 motivates the following:

Problem 3 (Parsimonious Founder Set) Given a set of haplotypes $\mathcal{H}$, find a founder set $\mathcal{F}$ that minimizes the total number of recombinations to generate all founder sequences from $\mathcal{H}$.

Constructing founder sets

In this section, we present a three-step solution to Problem 1 that is based on a network flow analysis of the *variation graph* over the input set of haplotypes. To this end, we introduce the notion of variation graphs and describe their construction for our specific setting. Subsequently, we define network flow and detail how a founder set can be derived. Our proposed network flow problem is subordinate to the Chinese Postman Problem on edge-colored multigraphs for which Gutin et al. proposed a polynomial algorithm [36]. Consequently, all other steps of our solution being polynomial, Problem 1 can be solved in polynomial time. However, we propose an integer linear program in lieu of Gutin et al's impractical algorithm. Then, in Section Results we show feasibility of our approach in experiments on simulated variation graphs and an exemplar biological data set.

Variation graph construction. We now address the construction of variation graph $G_{\mathcal{H}} = (V, E \cup \overrightarrow{E})$ from a given set of haplotypes $\mathcal{H}$. Graph $G_{\mathcal{H}}$ is an undirected edge-colored multigraph where each edge can have one of two colors corresponding to their membership in edge sets E and $\overrightarrow{E}$. In constructing $G_{\mathcal{H}}$, each marker m of the universe of forward-oriented markers $\mathcal{M}$ is represented by a tuple of its *extremities* $(m^{\mathrm{t}}, m^{\mathrm{h}})$ also called "*tail*" and "*head*" of m, respectively, and its reverse-oriented counterpart $\overline{m}$ is represented as $(m^{\mathrm{h}}, m^{\mathrm{t}})$. (Note that our notation is based on common practice of illustrating markers as arrows, that, in natural reading direction, face from left, i.e., tail of the arrow, to right, i.e., head of the arrow.) Node set V of graph $G_{\mathcal{H}}$ corresponds to the set of all marker extremities, and each marker $m \in \mathcal{M}$ gives rise to one *marker edge* $\{m^{\mathrm{t}}, m^{\mathrm{h}}\} \in \overrightarrow{E}$. Further, any two (not necessarily distinct) nodes $m_1^b, m_2^c \in V$ are connected by one *adjacency edge* $\{m_1^b, m_2^c\} \in E$ if they occur in one of the haplotypes either in forward or reverse order. More formally, there is an adjacency edge $\{m_1^b, m_2^c\} \in E$ if and only if there exists a sequence $A \in \mathcal{H} \cup \overline{\mathcal{H}}$ with $A = ..m_1 m_2..$ such that $m_1 = (m_1^a, m_1^b)$, $m_2 = (m_2^c, m_2^d)$ and $\{a, b\} = \{c, d\} = \{\mathrm{t}, \mathrm{h}\}$.

Example 2 Let $H_1 = \mathtt{s}\overline{\mathtt{1}}\mathtt{23}\overline{\mathtt{43}}\mathtt{S}$, $H_2 = \mathtt{s111234}\overline{\mathtt{3}}\mathtt{S}$, $H_3 = \mathtt{s}\overline{\mathtt{123432343}}\mathtt{S}$, and $H_4 = \mathtt{s}\overline{\mathtt{12}}\mathtt{S}$, then the variation graph $G_{\mathcal{H}}$ of $\mathcal{H} = \{H_1, H_2, H_3, H_4\}$ is as illustrated in Fig. 2, with marker edges drawn in gray and adjacency edges in black.

Proposition 1 *Let $G_{\mathcal{H}}$ be the variation graph of haplotypes $\mathcal{H}$, and $\mathcal{X}$ the set of all walks between terminal markers s^{t} and S^{h} in $G_{\mathcal{H}}$ with edges alternating between E and $\overrightarrow{E}$, then* span $(\mathcal{H}) = \mathcal{X}$.

Proof $\subseteq$ Observe that no recombination can create a new pair of consecutive markers $m_1 m_2$ that is not contained in any sequence $A \in \mathcal{H} \cup \overline{\mathcal{H}}$. Therefore, each haplotype $B \in$ span $(\mathcal{H})$ is a succession of consecutive markers drawn from sequences in $\mathcal{H} \cup \overline{\mathcal{H}}$, i.e., B can be delineated in $G_{\mathcal{H}}$ by following adjacency edges corresponding to its succession of consecutive markers.

$\supseteq$ Given an alternating walk $X = (s^{\mathrm{t}}, s^{\mathrm{h}}, \ldots, S^{\mathrm{t}}, S^{\mathrm{h}}) \in \mathcal{X}$, we show how to express X as a series of recombination events:

(a) Pick some haplotype $A \in \mathcal{H}$ and initialize $i \leftarrow 2$;
(b) Let $B \in \mathcal{H} \cup \overline{\mathcal{H}}$ be a sequence such that for some position j, $B[j..j+1] = m_1 m_2$ with $m_1 = X[i-1..i]$ and $m_2 = X[i+1..i+2]$. Then $A \leftarrow \chi(A, B, i/2, j)$.
(c) Increase i by 2 and repeat step **b** unless $i = |X| - 2$.

Observe that by construction of the variation graph $G_{\mathcal{H}}$, a suitable sequence $B \in \mathcal{H} \cup \overline{\mathcal{H}}$ must exist in each iteration of step **b**. □

Defining flows on variation graphs. We determine a minimum set of founder sequences by solving a network flow

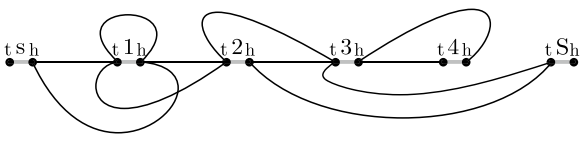


Fig. 2 Illustration of variation graph from Example 2

problem in variation graph $G_{\mathcal{H}}$ where flow is allowed to travel along adjacency edges in either direction. Algorithm 1 describes the network flow problem. Each node is associated with two capacities corresponding to incoming and outgoing flow

time and which we describe below, adapted to our circumstances. The idea is to perform a random walk in the graph from source to sink or within a cycle, thereby consuming flow along adjacency edges until all flow is

Algorithm 1 A network flow solution to Problem 1.

Objective:

$$\texttt{Minimize} \sum_{u,v \in V} \phi(u,v)$$

Flow capacities:

(F.01)	$i(v)$	$:= \sum_{u \in V} \phi(u,v)$	$\forall\, v \in V \setminus \{s^{\mathrm{t}}, S^{\mathrm{h}}\}$	*(incoming flow)*
(F.02)	$o(v)$	$:= \sum_{u \in V} \phi(v,u)$		*(outgoing flow)*
(F.03)	$o(s^{\mathrm{t}}) := i(S^{\mathrm{h}})$	$:= 0$		*(flow direction $s^{\mathrm{t}} \to S^{\mathrm{h}}$)*
	$i(s^{\mathrm{t}}), o(S^{\mathrm{h}})$	≥ 0		

Constraints:

(C.01)	$\phi(u,v)$	$\in \mathbb{N}$	$\forall\, u,v \in V$	*(constrain flow to integer)*
(C.02)	$\phi(u,v) = \phi(v,u)$	$= 0$	$\forall\, \{u,v\} \notin E$	*(constrain travel of flow)*
(C.03)	$\phi(u,v) + \phi(v,u)$	≥ 1	$\forall\, \{u,v\} \in E$	*(flow coverage)*
(C.04)	$i(m^{\mathrm{t}})$	$= o(m^{\mathrm{h}})$	$\forall\, m \in \mathcal{M}$	*(flow conservation)*
	$i(m^{\mathrm{h}})$	$= o(m^{\mathrm{t}})$		

We then find a non-negative flow $\phi : V \times V \to \mathbb{N}$ such that the total flow $\sum_{u,v \in V} \phi(u,v)$ of graph $G_{\mathcal{H}}$ is minimized and satisfies constraints. Note that the flow can travel in both directions of an edge $\{u,v\} \in E$ and that $\phi(u,v) = \phi(v,u)$ does not hold true in general.

Example 2 (cont'd) The drawing in Fig. 3 illustrates a flow solution on variation graph $G_{\mathcal{H}}$, with the direction and amount of flow along adjacency edges indicated by labeled arrowed arcs.

Deriving haplotypes from flows. By applying the Flow Decomposition Theorem [37, p. 80f], any *flow*, i.e., solution to the above-specified constraints, is decomposable into a set of alternating paths going from source s^{t} to sink S^{h} and a set of alternating cycles. Ahuja et al. [37] give a simple and efficient algorithm that does so in polynomial

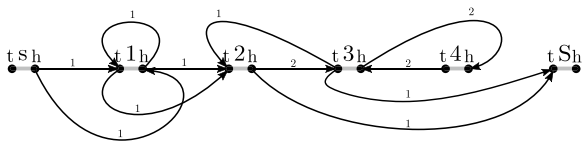


Fig. 3 Network flow solution on variation graph $G_{\mathcal{H}}$ of Example 2

depleted. The proof of the algorithm remains unchanged to that given by Ahuja et al., thus is not repeated here.

1 Set $u \leftarrow s^{\mathrm{t}}$.
2 Each node is adjacent to exactly one other node through a marker edge. Setting out from current node u, traverse this incident marker edge to some node v, choose any neighbor w of v for which $\phi(v,w) > 1$. Follow the adjacency edge to v and decrease the flow $\phi(v,w)$ by 1. Set $u \leftarrow w$.
3 As long as $u \neq S^{\mathrm{t}}$ do as follows: if u has been visited in the traversal before, then extract the corresponding alternating cycle from the recorded sequence and report it. Proceed with the traversal by repeating Step 2.
4 However, if $u = S^{\mathrm{t}}$, follow the marker edge to S^{h} and report the recorded sequence as a path.
5 If s^{h} is incident with edges with positive flow, proceed with Step 1. Otherwise, there still might be strictly positive flow remaining in the graph corresponding to unreported cycles. In that case, pick any node $u \leftarrow m^a$ such that for some node w, $\phi(m^b, w) > 0$, $\{a,b\} = \{\mathrm{t},\mathrm{h}\}$ and $m \in \mathcal{M}$, and proceed with Step 2 in order to report the next cycle.

Example 2 (cont'd) The components of the flow solution on variation graph $G_{\mathcal{H}}$ comprise two cycles C1 and C2, and two (s^t, S^h)-paths P1 and P2 are illustrated in Fig. 5.

What remains is the integration of cycles into walks that then correspond to the haplotypes of the founder set. The integration is facilitated by a graph structure, the *component graph*. The component graph $G' = (V', E', l)$ is an edge-labeled, directed multigraph, where, in its initial construction, each alternating (s^t, S^h)-path and each cycle reported during flow decomposition is represented by a distinct node of V'. In the component graph G', each cycle c of the flow decomposition sharing one or more markers with another component c' is connected by one or more directed edges (c, c') to that component, with each edge's label $l(c, c')$ corresponding to one distinct shared marker, oriented according to the their succession in c (which may not be the same as in c'). The component graph is then successively deconstructed until empty as follows:

1 Remove and report all (s^t, S^h)-walks with in-degree 0 from node set V'. Note that by construction, (s^t, S^h)-walks have out-degree 0, i.e., those with in-degree 0 are singleton in G'.
2 Pick a cycle $c \in V'$ with in-degree 0, or, if none such exists, any arbitrary cycle $c \in V'$.
3 Pick an outgoing edge $(c, c') \in E'$ such that c' is a (s^t, S^h)-walk. If no such c' exists, c is only adjacent to cycles, out of which one c' is picked arbitrarily. Let $(m^a, m^b) \leftarrow l(c, c')$, $\{a, b\} = \{t, h\}$. If marker m is embedded in c' in same orientation, i.e. $c' = ..m^a m^b..$, then linearize c in m, i.e., $c = m^b c_1..c_{k-1}m^a$, and integrate it into c' such that $c' \leftarrow ..m^a m^b c_1..c_{k-1}m^a m^b..$. Otherwise, integrate the reversed linearization of c, i.e, $c' \leftarrow ..m^b m^a c_{k-1}..c_1 m^b m^a..$. Remove cycle c and its outgoing edges from component graph G'.
4 Proceed with step 1 until no more components remain and all (s^t, S^h)-walks are reported.

The search for components with in-degree 0 can be efficiently implemented through preorder traversal of G'. Note that each cycle must have at least one outgoing edge and that ultimately all cycles *must be* integrable into a (s^t, S^h)-walk, otherwise this would imply that $G_{\mathcal{H}}$ contains a disconnected, circular component that is not reachable by an alternating path from source s^t to sink S^h, thus contradicting the correctness of $G_{\mathcal{H}}$'s construction. The reported (s^t, S^h)-walks represent the wanted haplotypes of a founder set.

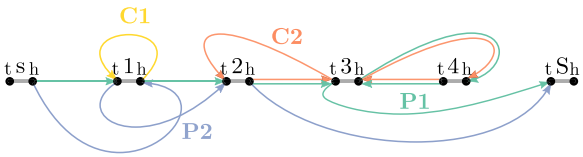


Fig. 4 Component graph of components C1, C2, P1, and P2 (left) and a founder set of $\mathcal{H}$ (right) from Example 2

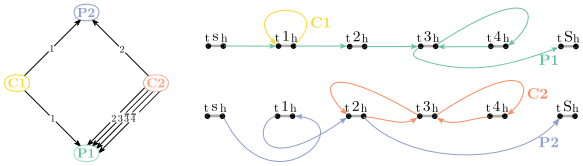


Fig. 5 Components of flow solution on variation graph $G_{\mathcal{H}}$ of Example 2

Example 2 (cont'd) The plots in Fig. 4 depict the component graph of components C1, C2, P1, and P2 (left) and the final two (s^t, S^h)-walks that collectively represent a founder set of $\mathcal{H}$ (right).

We define the multiplicity of a consecutive marker pair $m_1 m_2$, for any $m_1, m_2 \in \mathcal{M} \cup \overline{\mathcal{M}}$, as the number of times it appears as segment in forward or reverse order in a set of sequences $\mathcal{X}$ and introduce the following function for its retrieval:

$$\begin{aligned}\mu_{\mathcal{X}}(m_1, m_2) &= |\{(A, i) \mid A[i..i+1] \\ &= m_1 m_2 \text{ or } A[i..i+1] \\ &= \overline{m_2 m_1} : A \in \mathcal{X}\}|\end{aligned}$$

Theorem 1 *Let $\mathcal{F} \subseteq$ span $(\mathcal{H})$, $\mathcal{F}$ is a solution to Problem 1 if and only if $\mu_{\mathcal{F}}$ corresponds to a minimum network flow in $G_{\mathcal{H}}$.*

Proof $\Rightarrow$ Any flow of variation graph $G_{\mathcal{H}} = (V, E)$ is decomposable into a set of haplotypes $\mathcal{X}$, as demonstrated above. Observe that the above-listed flow constraints enforce the derived haplotypes $\mathcal{X}$ to cover the entire graph $G_{\mathcal{H}}$ and consequently $G_{\mathcal{X}} = G_{\mathcal{H}}$. This implies that span $(\mathcal{X})$ = span $(\mathcal{H})$. Further, the total number of consecutive markers in a haplotype sequence A equals $|A| - 1$ and therefore solutions to the specified network flow problem minimize quantity $\sum_{u,v \in V} \phi(u, v) = \sum_{A \in \mathcal{X}} |A| - |\mathcal{X}|$. This is equivalent to minimizing $\sum_{A \in \mathcal{X}} |A|$, because it is not possible to reduce the founder set size by concatenating two or more founder sequences without increasing the number of consecutive markers by an equal amount. Conversely, in the network flow specification, the sink node has no outgoing flow to the source node and therefore any founder

set derived by a flow solution cannot be reduced by concatenation.

⇐We show that every founder set is also a solution to the specified minimum network flow problem. Assume that $\mathcal{F}$ is a founder set of haplotypes $\mathcal{H}$ and observe that multiplicities $\mu_{\mathcal{F}}$ correspond to a valid flow ϕ in $G_{\mathcal{H}}$. Now assume that there exists another flow ϕ' such that $\sum_{u,v\in V}\phi'(u,v) < \sum_{u,v\in V}\phi(u,v) = \sum_{A\in\mathcal{F}}|A| - |\mathcal{F}|$. Then, following the algorithm above, ϕ' can be decomposed into haplotype set $\mathcal{F}'$ such that $\sum_{A\in\mathcal{F}'}|A| - |\mathcal{F}'| < \sum_{A\in\mathcal{F}}|A| - |\mathcal{F}|$, contradicting the premise that $\mathcal{F}$ is a solution to Problem 1. □

Counting recombinations in founder sequences

We now provide a general algorithm for solving Problem 2. We show how this algorithm can be implemented to scale linearly with the input set of terminal sequences $\mathcal{T}$ and query sequence Q in time and space by utilizing generalized suffix trees. Supplementary Note N2 further describes a solution based on suffix arrays that has the same asymptotic runtime and space guarantees, but is considered more practical. Our approach builds on the concept that each terminal sequence Q that can be generated from set $\mathcal{T}$ is segmentable into a set of overlapping segments, where each such segment corresponds to a segment in a terminal sequence of $\mathcal{T}$. We call these segments *$\mathcal{T}$-blocks* for the remainder of this manuscript.

Lemma 2 *Q can be generated from terminal sequences $\mathcal{T}$ if and only if Q is segmentable into a sequence of overlapping $\mathcal{T}$-blocks $\mathcal{D} = \{Q[..i_1], Q[i_1..i_2], .., Q[i_n..]\}$, with $1 < i_1 < .. < i_n < |Q|$, i.e., for each $D \in \mathcal{D}$, $\exists\, A \in \mathcal{T}\cup\overline{\mathcal{T}}$ with $D \lhd A$.*

Proof ⇒ If Q can be generated from $\mathcal{T}$ then there exists a series of recombinations $Q_1 \leftarrow \chi(A_1, A_2, i_1, k_1)$, $Q_2 \leftarrow \chi(Q_1, A_3, i_2, k_2)$,.., $Q_m \leftarrow \chi(Q_{m-2}, A_m, i_m, k_m)$ such that $A_1, .., A_m \in \mathcal{T}\cup\overline{\mathcal{T}}$ and $Q_m = Q$. Consequently, Q can be segmented into the set of overlapping $\mathcal{T}$-blocks $\mathcal{D} = \{A_1[..i_1], A_2[k_1..k_1 + i_2 - i_1], .., A_m[k_m..]\}$. ⇐ For a given segmentation $\mathcal{D} = \{D_1, .., D_n\}$, Q is generated by a series of recombinations from $\mathcal{T}$ as follows: Let $Q_1[..i_1] = D_1$, $Q_1 \in \mathcal{T}\cup\overline{\mathcal{T}}$; For each x in $2..n$, $Q_x \leftarrow \chi(Q_{x-1}, A_x, i_x, k)$ where $A_x[k..k + i_x - i_{x-1}] = D_x$, $A_x \in \mathcal{T}$; Observe that $Q_n = Q$. □

Finding a minimum $\mathcal{T}$-block segmentation is equivalent to computing the minimum number of recombinations—the former differs in size from the latter only by an increment of 1. The recursive function $R_{\mathcal{T}} : \Theta \to \mathbb{N}$ defined below calculates the number of recombinations to generate query sequence Q from terminal sequences $\mathcal{T}$ by moving from one maximal $\mathcal{T}$-block of Q to the next. To this end, we define $L_{\mathcal{T}}(Q)$ as the length of the longest prefix of Q that is a $\mathcal{T}$-block, i.e., $L_{\mathcal{T}}(Q) := \arg\max_k \left\{Q[..k] \lhd A \mid A \in \mathcal{T}\cup\overline{\mathcal{T}}\right\}$.

$$R_{\mathcal{T}}(Q) = \begin{cases} 0 & \text{if } |Q| = L_{\mathcal{T}}(Q) \\ \infty & \text{else if } L_{\mathcal{T}}(Q) \leq 1 \\ 1 + R_{\mathcal{T}}(Q[L_{\mathcal{T}}(Q)..]) & \text{otherwise} \end{cases} \tag{1}$$

$R_{\mathcal{T}}(Q) = \infty$ indicates that Q cannot be generated from $\mathcal{T}$. We prove that the algorithm is optimal, i.e., computes the minimum number of recombinations:

Proof We prove this by induction over the number of recombinations identified by Eq. 1. Note that the total number of recombinations is bounded by the length of query Q. We show that for every k with $0 \leq k < |Q|$ that $R_{\mathcal{T}}(Q[..i_k])$ reports the minimum number of recombinations for sequence $Q[..i_k]$.

(IB) In iteration $k = 0$, $R(\cdot)$ receives the full-length query sequence Q and chooses the longest prefix of query Q that is a $\mathcal{T}$-block. It is clear that this is an optimal choice, since choosing a smaller prefix can only increase the number of recombinations. Note that if $l \leq 1$, Q cannot be generated from $\mathcal{T}$ and $R(Q)$ returns ∞.

(IS) Let $Q[i_{k-1}..i_k]$ be the $\mathcal{T}$-block identified in the k-th recurrence of $R_{\mathcal{T}}$ with the current query sequence being $Q[i_k..]$. In step $k+1$, $R_{\mathcal{T}}$ will again identify the segment $D = Q[i_k..i_k + l]$ of maximal length l that is a $\mathcal{T}$-block.
Let us now claim that there is an shorter sequence of $\mathcal{T}$-blocks $Q[j_1..j_2], Q[j_2, j_3], \ldots Q[j_{k-1}, j_k]$ and $i_k + l = j_k$, as illustrated in Fig. 6. Then there must be some $0 \leq k^\star \leq k$ for which $j_{k^\star} > i_{k^\star}$. But if there were indeed a $\mathcal{T}$-block $Q[j_{k^\star-1}..j_{k^\star}]$, then $Q[i_{k^\star-1}..j_{k^\star}]$ is a suffix of $Q[j_{k^\star-1}..j_{k^\star}]$ and that would be the longest common prefix chosen by $R(\cdot)$ in iteration $i_{k^\star-1}$, contradicting the definition of $R(\cdot)$. Therefore, a shorter sequences of $\mathcal{T}$-blocks cannot exist. □

The algorithm can efficiently count recombinations by utilizing the *suffix tree* data structure [38]. To this end, the suffix tree is constructed on sequence $T\$$ corresponding to a concatenation of terminal sequences $\mathcal{T}\cup\overline{\mathcal{T}}$ of any given order, terminated by sentinel "$". In doing so, we assume that terminal markers $s, \bar{s}, S$, and $\bar{S}$ abide lexicographic order $\$ < \{S, \bar{s}\} < \{s, \bar{S}\} < m\ \forall\, m \in \mathcal{M}\cup\overline{\mathcal{M}}$. Suffix trees can be constructed in linear time and space [39], and matching substrings in $T\$$ can be performed in

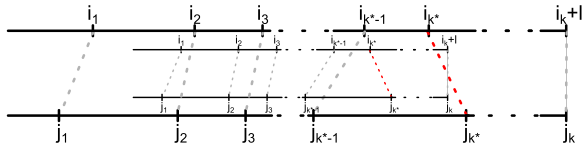


Fig. 6 Illustration of the contradictory claim a shorter sequence of $\mathcal{T}$-blocks can be constructed than found by Eq. 1. The red dashed line indicates the contradictory situation that $i_{k^\star} < j_{k^\star}$. In that case $Q[i_{k^\star-1}..j_{k^\star}]$ would have been chosen as longest $\mathcal{T}$-block in recursion step $k^\star - 1$

time linear to the length of the matching. To assess the time complexity of the recursion, observe that $R_{\mathcal{T}}(Q)$ is recursed at most $|Q| - 1$ times, if all $\mathcal{T}$-blocks have length 2. We conclude:

Theorem 3 *Problem 2 is solvable in $O(|\mathcal{T}| + |Q|)$ time and space.*

Minimizing recombinations in founder sequences

We now present an algorithm towards solving Problem 3, i.e., the problem of finding a founder set that minimizes the number of recombinations needed for its construction from a given set of haplotypes $\mathcal{H}$. Solving this problem requires the simultaneous computation of solutions to both the Founder Set and the Recombination Count problem and constitutes in combing through an exponentially large search space. We simplify the problem by presuming that the multiplicities of consecutive marker pairs in a solution to the Parsimonious Founder Set Problem are also optimal under the Founder Set problem. In other words, our approach is exact under the assumption that the overall multiplicity of each pair of consecutive markers in a founder set that is a solution to Problem 3 is known, yet the pair's particular orientation and location in the founder sequences are not. To this end, we presume a function $\hat{\mu}_{\mathcal{F}}(m_1, m_2)$ acting as oracle for the overall multiplicity of any given pair of consecutive oriented markers $m_1, m_2 \in \mathcal{M} \cup \overline{\mathcal{M}}$ in a solution $\mathcal{F}$ to Problem 3. More specifically, $\hat{\mu}_{\mathcal{F}}(m_1, m_2)$ reports the total number of occurrences of $m_1 m_2$ and $\overline{m_2 m_1}$ in founder set $\mathcal{F}$. Note that our experiments directly use the results of Problem 1 as input for Problem 3, i.e., $\hat{\mu}_{\mathcal{F}}(m_1, m_2)$ reports the number of occurrences of (m_1, m_2) in a solution to Problem 1. This makes our experimental solutions to Problem 3 heuristic. In addition, we make use of function $\hat{\gamma}_{\mathcal{F}}(m) := \sum_{m' \in \mathcal{M} \cup \overline{\mathcal{M}}} \hat{\mu}_{\mathcal{F}}(m, m')$ to retrieve the multiplicity of any marker $m \in \mathcal{M} \cup \overline{\mathcal{M}}$. Note that $\hat{\mu}_{\mathcal{F}}$ and $\hat{\gamma}_{\mathcal{F}}$ are symmetric with respect to the relative orientation of markers, $\hat{\mu}_{\mathcal{F}}(m_1, m_2) = \hat{\mu}_{\mathcal{F}}(\overline{m_2}, \overline{m_1})$ and $\hat{\gamma}_{\mathcal{F}}(m) = \hat{\gamma}_{\mathcal{F}}(\overline{m})$. Our solution makes use of the *flow graph* that is defined in the subsequent paragraph. We calculate a matching in the flow graph that describes a set of founder sequences, each corresponding to a succession of segments of haplotypes $\mathcal{H}$. The objective of the matching is to minimize the total number of $\mathcal{H}$-blocks across all founder sequences which is equivalent to minimizing the number of recombinations for their construction from haplotype set $\mathcal{H}$.

Flow graph construction. The flow graph $G_{\mathcal{H},\hat{\mu}_{\mathcal{F}}} = (V_{\hat{\mu}_{\mathcal{F}}}, E_{\hat{\mu}_{\mathcal{F}}} \cup \overrightarrow{E_{\hat{\mu}_{\mathcal{F}}}})$ is a directed edge-colored multigraph with adjacency edges $E_{\hat{\mu}_{\mathcal{F}}}$ and marker edges $\overrightarrow{E_{\hat{\mu}_{\mathcal{F}}}}$, where each marker extremity m^a with $m \in \mathcal{M}$ and $a \in \{\mathrm{t}, \mathrm{h}\}$, gives rise to $2 \cdot \hat{\gamma}_{\mathcal{F}}(m)$ elements in node set $V_{\hat{\mu}_{\mathcal{F}}}$, representing $\hat{\gamma}_{\mathcal{F}}(m)$ many *in* (i) and $\hat{\gamma}_{\mathcal{F}}(m)$ many *out* (o) nodes. Hence, each node in the flow graph is represented by a triple of the form $\{\mathrm{i}, \mathrm{o}\} \times \mathcal{M}^{\mathrm{t}} \cup \mathcal{M}^{\mathrm{h}} \times \mathbb{N}$ with the complete vertex set being $V_{\hat{\mu}_{\mathcal{F}}} = \{(\mathrm{i}, m^a, x) \mid m \in \mathcal{M}, a \in \{\mathrm{t}, \mathrm{h}\}, x \in 1..\hat{\gamma}_{\mathcal{F}}(m)\} \cup \{(\mathrm{o}, m^a, x) \mid m \in \mathcal{M}, a \in \{\mathrm{t}, \mathrm{h}\}, x \in 1..\hat{\gamma}_{\mathcal{F}}(m)\}$. Each out node $u \in V_{\hat{\mu}_{\mathcal{F}}} \setminus (\{(\mathrm{i}, S^{\mathrm{h}}, x) \mid 1..\hat{\gamma}_{\mathcal{F}}(S)\} \cup \{(\mathrm{o}, s^{\mathrm{t}}, x) \mid 1..\hat{\gamma}_{\mathcal{F}}(s)\})$ is incident with *one and only one* directed adjacency edge (u, v) connecting u to some in node v thereby realizing one occurrence of its representing pair of consecutive oriented markers in a founder sequence. Conversely, each forward-oriented marker $m \in \mathcal{M}$ contributes $\hat{\gamma}_{\mathcal{F}}(m)^2$ many directed marker edges that connect in/tail nodes with out/head nodes, i.e., $\{((\mathrm{i}, m^{\mathrm{t}}, x), (\mathrm{o}, m^{\mathrm{h}}, y)) \mid x, y \in 1..\hat{\gamma}_{\mathcal{F}}(m)\}$. Analogously, each reverse-oriented marker $\overline{m} \in \overline{\mathcal{M}}$ contributes $\hat{\gamma}_{\mathcal{F}}(m)^2$ many in/head-to-out/tail-directed marker edges $\{((\mathrm{i}, m^{\mathrm{h}}, x), (\mathrm{o}, m^{\mathrm{t}}, y)) \mid x, y \in 1..\hat{\gamma}_{\mathcal{F}}(m)\}$.

Example 2 (cont'd) Fig. 7 visualizes the flow graph $G_{\mathcal{H},\hat{\mu}_{\mathcal{F}}}$ for the given set of haplotypes $\mathcal{H} = \{\mathtt{s}\overline{1}23\overline{43}\mathtt{S}, \mathtt{s}1112343\mathtt{S}, \mathtt{s}\overline{1}23\overline{432}\overline{43}\mathtt{S}, \mathtt{s}\overline{1}2\mathtt{S}\}$ and a given $\hat{\mu}_{\mathcal{F}}$.

Graph decomposition. A perfect matching of marker edges in flow graph $G_{\mathcal{H},\hat{\mu}_{\mathcal{F}}}$ produces a set of alternating walks and alternating cycles through $G_{\mathcal{H},\hat{\mu}_{\mathcal{F}}}$, yet only half of the graph is eligible to form a solution to Problem 3. More precisely, for each marker $m \in \mathcal{M}$, exactly half of the number of its associated nodes in $V_{\hat{\mu}_{\mathcal{F}}}$ must be *saturated*, i.e., incident with a matching edge. The other half as well as their incident edges must remain unsaturated. Further, we aim to admit only matchings that consist entirely of alternating $((\mathrm{i}, s^{\mathrm{t}}, x), (\mathrm{o}, S^{\mathrm{h}}, y))$-walks, because only those correspond to valid haplotypes of span $(\mathcal{H})$.

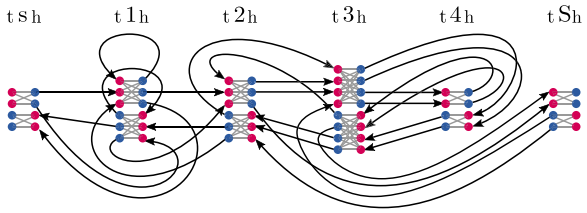
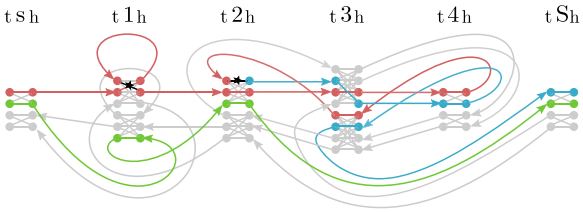


Fig. 7 Flow graph $G_{\mathcal{H},\hat{\mu}_{\mathcal{F}}}$ of Example 2. In nodes and out nodes are highlighted in red and blue, respectively. For clarity, the direction of marker edges (gray edges; directed from in to out node) is omitted in the illustration



$F1 : s\bar{1}2S \qquad F2 : s112343\bar{2}34\bar{3}S$

Fig. 8 Solution to Algorithm 2 for $G_{\mathcal{H},\hat{\mu}_{\mathcal{F}}}$ for Example 2

At last, we aim to assign to each saturated node $v \in V_{\hat{\mu}_{\mathcal{F}}}$ a position in some haplotype A of given haplotype set $\mathcal{H}$. That way, we are able to determine whether the incident adjacency edge serves as continuation of the associated haploblock of A, or whether the incident saturated marker edge implies a recombination between two distinct $\mathcal{H}$-blocks.

The *Integer Linear Program* shown in Algorithm 2 implements the above-stated constraints.

Example 2 (cont'd) Fig. 8 illustrates a matching that is solution to Algorithm 2 for $G_{\mathcal{H},\hat{\mu}_{\mathcal{F}}}$. The founder sequences are spelled out on the bottom, colored by haplotype (red, blue and green for haplotypes 2, 3 and 4 respectively). Unsaturated nodes and edges are grayed out, haplotype assignments implied by colored paths. The solution features two recombinations, marked by "$\star$" along their associated marker edges.

Objective. The ILP maximizes the sum over all $\mathtt{t}$ variables, which corresponds to finding a set of founder sequences that has a maximum number of marker pairs m_1m_2 associated with consecutive positions in any of the haplotypes $\mathcal{H}$. Conversely, any marker pair that is not linked to a position in a haplotype of $\mathcal{H}$ represents a recombination event.

Matching constraints. Each edge $(u,v) \in E_{\hat{\mu}_{\mathcal{F}}} \cup \overrightarrow{E_{\hat{\mu}_{\mathcal{F}}}}$ and node $w \in V_{\hat{\mu}_{\mathcal{F}}}$ of flow graph $G_{\mathcal{H},\hat{\mu}_{\mathcal{F}}}$ is associated with binary variables of $\mathtt{x}(u,v)$ and $\mathtt{y}(w)$, respectively, that determine their saturation in a solution (cf. domains D.1 and D.2). Constraint C.01 ensures that each saturated marker edge is incident with saturated nodes. Perfect matching constraints, i.e., constraints that impose each saturated node being incident with exactly one marker edge, are implemented by constraint C.02. Similarly, constraint C.03 ensures that an adjacency edge is saturated if and only if its incident nodes are saturated. In other words, constraints C.01-C.03 together ensure that each component of the saturated graph corresponds to an alternating path or cycle component (the latter being prohibited by further constraints). The following two constraints C.04 and C.05 control the overall size of the saturated graph. In doing so, they ensure that, in a solution to Problem 3, the number of saturated nodes and adjacency edges matches the postulated multiplicity of markers $\hat{\gamma}_{\mathcal{F}}(m)$, $m \in \mathcal{M} \cup \overline{\mathcal{M}}$, and pairs of consecutive markers $\hat{\mu}_{\mathcal{F}}(m_1,m_2), m_1,m_2 \in \mathcal{M} \cup \overline{\mathcal{M}}$, respectively.

Path constraints. Constraints C.05-C.08 force each component of the saturated graph to start and end in nodes associated with source s^{t} and sink S^{h}, respectively, thereby ruling out any cycles. To this end, they make use of a set of integer variables $\mathtt{f}(v)$ over all vertices $v \in V_{\hat{\mu}_{\mathcal{F}}}$ (cf. Domain D.03) that define an increasing flow within each saturated component that is bounded by constant T corresponding to the total flow of the graph, i.e., $T := \sum_{m \in \mathcal{M}} \hat{\gamma}_{\mathcal{F}}(m)$. In each saturated marker edge, the flow is increased by 1 while along each adjacency edge, flow is kept constant. This prevents the formation of saturated cycles, because their flow would be infinite. Lastly, constraint C.08 preclude paths from starting in S^{h} or ending in s^{t}, leaving only one option for any saturated component open, that is, the formation of a $(s^{\mathrm{t}}, S^{\mathrm{h}})$-path.

Haplotype assignment. Each node $v \in V_{\hat{\mu}_{\mathcal{F}}}$ in a solution to the ILP is associated with exactly one position $j \in 1..|A|$ in a haplotype A of $\mathcal{H}$, recorded by binary variables $\mathtt{c}(A[j],v)$.Q Moreover, any marker edge whose incident pair of nodes is associated with the same position of the same haplotype corresponds to a $\mathcal{H}$-block, i.e, no recombination within this marker has taken place. Each marker edge $(u,v) \in \overrightarrow{E_{\hat{\mu}_{\mathcal{F}}}}$ that is linked by the ILP solver to a position j in a haplotype $A \in \mathcal{H}$ contributes a score unit to the objective function. These score units are encoded by binary variables $\mathtt{t}(A[j],u,v)$ (cf. domain D.05). Constraint C.09 ensures that each marker is associated with exactly one position j in a haplotype A of set $\mathcal{H} \cup \overline{\mathcal{H}}$, while C.10 confines incident nodes of adjacency edges to represent a consecutive marker pair $A[j..j+1]$. At last, constraint C.11 allows $\mathtt{t}$ variables of marker edges to take on value 1 only if that marker edge is saturated and its incident nodes are associated with the same haplotype position.

Algorithm 2 An ILP solution to Problem 3.

Objective:

$$\texttt{Maximize} \sum_{\substack{(u,v)=((\mathrm{i},m^a,x),(\mathrm{o},m^b,x'))\in\overrightarrow{E_{\hat{\mu}_{\mathcal{F}}}},\\ A[j]=(m^a,m^b)}} \mathtt{t}(A[j],u,v)$$

Constraints:

(C.01)	$\mathtt{y}(u)+\mathtt{y}(v)\geq 2\ \mathtt{x}(u,v)$	$\forall\ (u,v)\in\overrightarrow{E_{\hat{\mu}_{\mathcal{F}}}}$
(C.02)	$\sum_{(u,v)\in\overrightarrow{E_{\hat{\mu}_{\mathcal{F}}}}} \mathtt{x}(u,v)=\mathtt{y}(u)$	$\forall$ in nodes $u\in V_{\hat{\mu}_{\mathcal{F}}}$
	$\sum_{(u,v)\in\overrightarrow{E_{\hat{\mu}_{\mathcal{F}}}}} \mathtt{x}(u,v)=\mathtt{y}(v)$	$\forall$ out nodes $v\in V_{\hat{\mu}_{\mathcal{F}}}$
(C.03)	$\mathtt{x}(u,v)=\mathtt{y}(u)$	$\forall\ (u,v)\in E_{\hat{\mu}_{\mathcal{F}}}$
	$\mathtt{x}(u,v)=\mathtt{y}(v)$	
(C.04)	$\sum_{x=1}^{\hat{\gamma}_{\mathcal{F}}(m)} \mathtt{y}(\mathrm{i},m^a,x)+\mathtt{y}(\mathrm{o},m^a,x)=\hat{\gamma}_{\mathcal{F}}(m)$	$\forall\ m\in\mathcal{M}, a\in\{\mathrm{t},\mathrm{h}\}$
(C.05)	$\sum_{\substack{x,x' \text{ s.t.}\\ ((o,m_1^b,x),\\ (\mathrm{i},m_2^c,x'))\in E_{\hat{\mu}_{\mathcal{F}}}}} \mathtt{x}((\mathrm{o},m_1^b,x),(\mathrm{i},m_2^c,x'))=\hat{\mu}_{\mathcal{F}}(m_1,m_2)$	$\forall\ (m_1^b,m_2^c)$ s.t. $\hat{\mu}_{\mathcal{F}}(m_1,m_2)>0$, $m_1=(m_1^a,m_1^b),\ m_2=(m_2^c,m_2^d)$, $\{a,b\}=\{c,d\}=\{\mathrm{t},\mathrm{h}\}$
(C.06)	$\mathtt{f}(u)=\mathtt{f}(v)$	$\forall\ (u,v)\in E_{\hat{\mu}_{\mathcal{F}}}$
(C.07)	$\mathtt{f}(v)-\mathtt{f}(u)+T\mathtt{x}(u,v)\leq T+1$	$\forall\ (u,v)\in\overrightarrow{E_{\hat{\mu}_{\mathcal{F}}}}$
(C.08)	$\mathtt{f}(v)=0$	$\forall v\in\{(\mathrm{o},s^{\mathrm{t}},x)\mid x\in 1..\hat{\gamma}_{\mathcal{F}}(s)\}\cup\{\mathrm{i},S^{\mathrm{h}},x)\mid x\in 1..\hat{\gamma}_{\mathcal{F}}(S)\}$
(C.09)	$\sum_{\substack{A\in\mathcal{H}\\ A[j]=m}} \mathtt{c}(A[j],v)=1$	$\forall\ v\in V_{\hat{\mu}_{\mathcal{F}}},\ v$ associated with extremities of marker m
(C.10)	$\mathtt{c}(A[j],(\mathrm{o},m_1^b,x))=$ $\mathtt{c}(A[j+1],(\mathrm{i},m_2^c,x'))$	$\forall\ ((\mathrm{o},m^b,x),(\mathrm{i},m^a,x'))\in E_{\hat{\mu}_{\mathcal{F}}}$, $A\in\mathcal{H}\cup\overline{\mathcal{H}}, i\in 1..\lvert A\rvert-1$, s.t. $A[j..j+1]=(m_1^a,m_1^b)(m_2^c,m_2^d)$
(C.11)	$\mathtt{x}(u,v)+\mathtt{c}(A[j],u)+\mathtt{c}(A[j],v)\geq 3\ \mathtt{t}(A[j],u,v)$	$\forall\ (u,v)=((\mathrm{i},m^a,x),(\mathrm{o},m^b,x'))\in\overrightarrow{E_{\hat{\mu}_{\mathcal{F}}}},\ A\in\mathcal{H}\cup\overline{\mathcal{H}}, i\in 1..\lvert A\rvert$, s.t. $A[j]=(m^a,m^b)$

Domains:

(D.01)	$\mathtt{x}(u,v)\in\{0,1\}$	$\forall\ (u,v)\in E_{\hat{\mu}_{\mathcal{F}}}\cup\overrightarrow{E_{\hat{\mu}_{\mathcal{F}}}}$
(D.02)	$\mathtt{y}(v)\in\{0,1\}$	$\forall\ v\in V_{\hat{\mu}_{\mathcal{F}}}$
(D.03)	$1\leq\mathtt{f}(v)\leq T$	$\forall\ v\in V_{\hat{\mu}_{\mathcal{F}}}$
(D.04)	$\mathtt{c}(A[j],(\mathrm{i},m^a,x))$, $\mathtt{c}(A[j],(\mathrm{o},m^a,x))$, $\mathtt{c}(A[j],(\mathrm{i},m^b,x))$, $\mathtt{c}(A[j],(\mathrm{o},m^b,x))\in\{0,1\}$	$\forall\ A\in\mathcal{H}\cup\overline{\mathcal{H}},\ j\in 1..\lvert A\rvert$, $A[j]=(m^a,m^b), x\in 1..\hat{\gamma}_{\mathcal{F}}(m)$
(D.05)	$\mathtt{t}(A[j],(\mathrm{i},m^a,x),(\mathrm{o},m^b,x'))\in\{0,1\}$	$\forall\ A\in\mathcal{H}\cup\overline{\mathcal{H}},\ j\in 1..\lvert A\rvert$, $A[j]=(m^a,m^b), x\in 1..\hat{\gamma}_{\mathcal{F}}(m)$

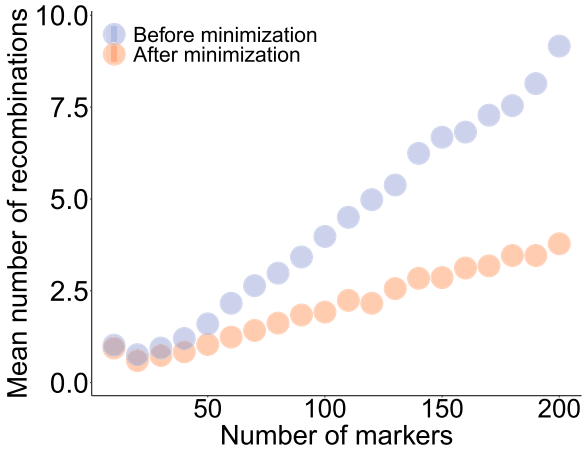


Fig. 9 *Mean number of recombinations by the size of the graph.* Experiments were ran with values ranging from 10 to 200 in for the number of markers, in increments of 10. The ratio of duplications and of inversions was fixed to 10%, and number of haplotypes to 10. Each colored dot represents the mean number of recombinations over 50 replicates for one parameter set, after random assignment trials (blue) and after optimization (red)

Results

We implemented our methods in the programming language Rust [40] and used Gurobi [41] as the solver. Our software is *open source* and publicly available online [42]. To run Algorithm 2 on a given set of haplotypes $\mathcal{H}$, we estimated the overall multiplicity $\hat{\mu}_{\mathcal{F}}(m_1, m_2)$ of pairs of consecutive markers $m_1 m_2$ from a network flow solution to Problem 1 on $\mathcal{H}$. Note that, because there is no guarantee that an optimal solution to Problem 3 exists that has also optimal flow under Problem 1, our approach does not guarantee exact solutions.

For benchmarking purposes, we ran Gurobi single-threaded and recorded wall clock time (in seconds) and *Proportional Set Size (PSS)* (in Megabytes (MB)) for memory usage. The choice of using PSS rather than measures such as *Resident Set Size (RSS)* or *Unit Set Size (USS)* is largely arbitrary, however all three measures were highly similar in all experiments and within 100 MB of each other at the extreme. Optimization time was capped at 30 min, beyond which the solver stops and returns its best-effort solution found thus far.

Experimental data

We benchmarked the performance of our algorithms by conducting experiments on both simulated data and a real-world data set. The former presumed a simulator, capable of generating haplotypes with duplicated and inverted markers that can produce intricate homologous recombinations while providing control over the degree of complexity. To this end, we implemented our own simulation tool that constructs a single haplotype sequence sampled at random to serve as seed. This seed sequence is adjustable by the following parameters: (i) number of distinct markers, i.e., the size of its variation graph, (ii) ratio of duplications, i.e., the number of additional edges inducing duplications in a walk of the graph, (iii) ratio of inversions, i.e., the proportion of inverted orientations within the set of duplications, and lastly (iv) the number of haplotypes that are input to subsequent founder set reconstruction. The latter are generated by performing random walks in the seed sequence's variation graph and retaining only those leading from source to sink. In doing so, our simulator does not report nor have knowledge of a true founder set. Our simulator, discussed in more detail in Supplementary Note N1, enables us to explore various parameterizations that match different situations in biological data.

One important point concerns co-optimality. Problems 1 and 3 do not guarantee a unique solution. In fact, the pool of co-optimal solutions is often large for both problems. One contributing factor to co-optimality are cycles that are shared across multiple haplotypes, because they can be integrated in different orders. Further, the solution does not provide any information that could enable one to generate all co-optimal solutions nor discern between them, making a measure of accuracy challenging, since there is no guarantee that the "correct" founder sequence(s) will be seen in any number of trials.

In addition to simulated data, we applied our methods on a biological data set from the human 1p36.13 locus described by Porubsky et al. [22] to demonstrate the computational performance on realistic instances.

Simulation experiments

To assess the impact of parameter configurations on the results, we ran a number of different experiments wherein all but one parameters are fixed. A reasonable choice of constants seemed to be 100 distinct markers, 10% of duplications, 10% of inversions and 10 haplotypes, motivated by our data on the 1p36.13 locus (8 markers, 68 haplotypes, 57% of duplications) and statistics compiled by Porubsky et al. [22] ($6-7$% duplications in the whole genome, <1% inversions).

Reduction in number of recombinations. To evaluate the efficacy of our solution to Problem 3, we compared the number of recombinations returned by Algorithm 2 to that in a solution obtained by our network flow algorithm for Problem 1. To this end, we set the output of Algorithm 1 against an implementation of a solution to Problem 2, described in further detail in Supplementary Note N2. Figure 9 summarizes the outcome of this experiment.

Overall, Algorithm 2 found a solution with fewer recombinations in all instances but a few where Gurobi returned barely best-effort solutions after reaching the time limit of 30 min, all of which exhibited a gap of at least 100%. The parameter settings in those cases were extremal.

Across all experiments, the mean estimated number of recombinations increased linearly by approximately 1.7 per 100 markers after minimization, compared to 4.5 per 100 without it. The values reached respectively 3.8 and 9.1 at 200 markers. The simulations here were carried out with a fixed number of haplotypes and ratios of duplications and of inversions. Results for experiments with other variable parameters are shown in Additional file 1: Figure S1.

Flow solution benchmark. Computing solutions with our network flow algorithm proved to be in almost all of our experiments near-instantaneous. By varying the number of distinct markers, the algorithm's performance begins to deteriorate only with very large instances beyond 100k distinct markers and becomes excruciating for instances above 1M markers. When varying other parameters, we fixed the number of distinct markers to 100k rather than 100. Under 100k markers, execution completed after a mean wall clock time of 3.4 ± 2.0 seconds. In 95% of all experiments, the solver's runtime was too short to make sufficient measurements for benchmarking memory usage; the maximum PSS for the remaining ones measured at 78 MB. Over the 100k mark, both the graph size and duplication ratio began to reduce performance, with an average runtime of 19.7 ± 8.7s. The ratio of inversions on the other hand did not affect performance (Suppl. Figure S3). We measured peak memory consumption at 758 MB across all conditions, which also occurred only at the very extremes of 100k distinct markers and a 100% ratio of duplications (Fig. 10).

Recombination minimization benchmark. As shown previously, Algorithm 2 successfully reduces the number of recombinations in solutions to Problem 1. However, its runtime increased dramatically with only moderate increments of any but one parameter of our simulator, the ratio of inversions, which did not play

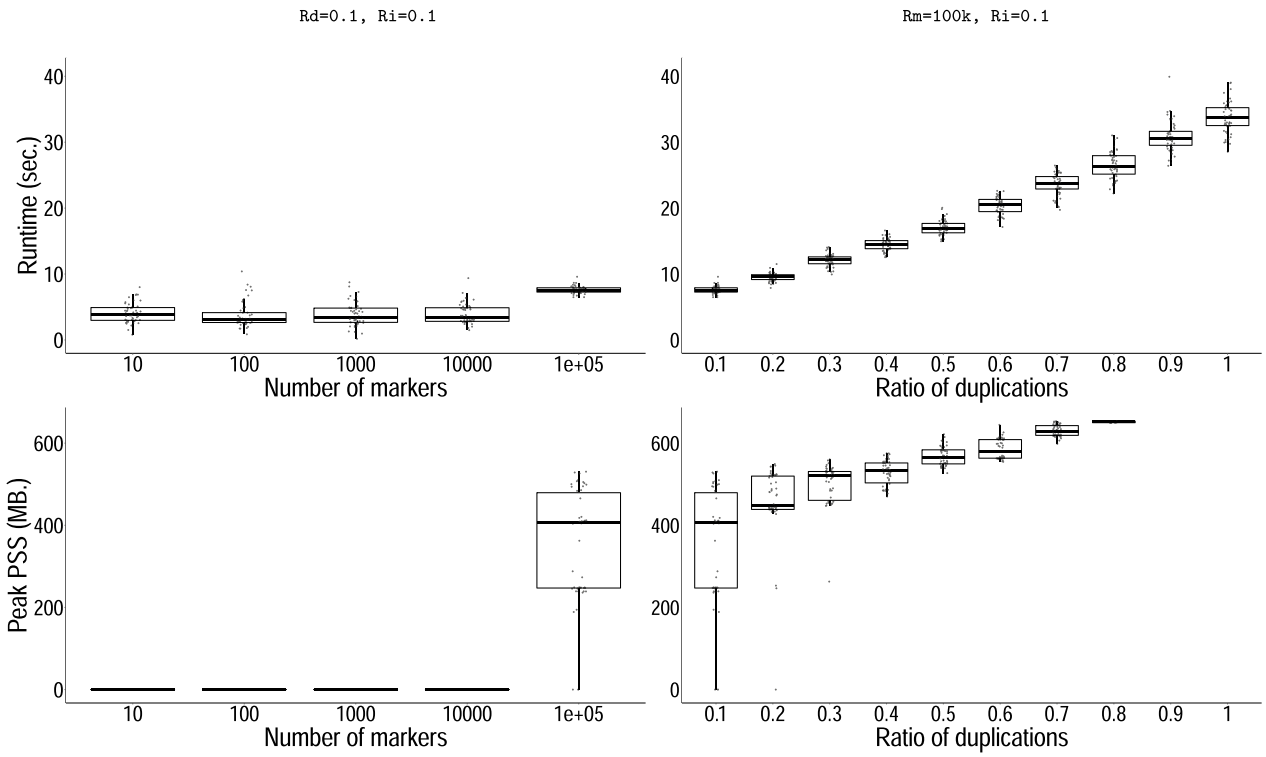


Fig. 10 *Problem 1, flow computational performance benchmarks.* Runtime in seconds (upper panels) and peak PSS in MB (lower panels), as a function of the number of markers (left) and of the ratio of duplications (right). For each experiment, the remaining parameters are fixed as indicated above. The abbreviations read as follows: *Nm* number of markers; *Rd* ratio of duplications; and *Ri*, ratio of inverted duplications

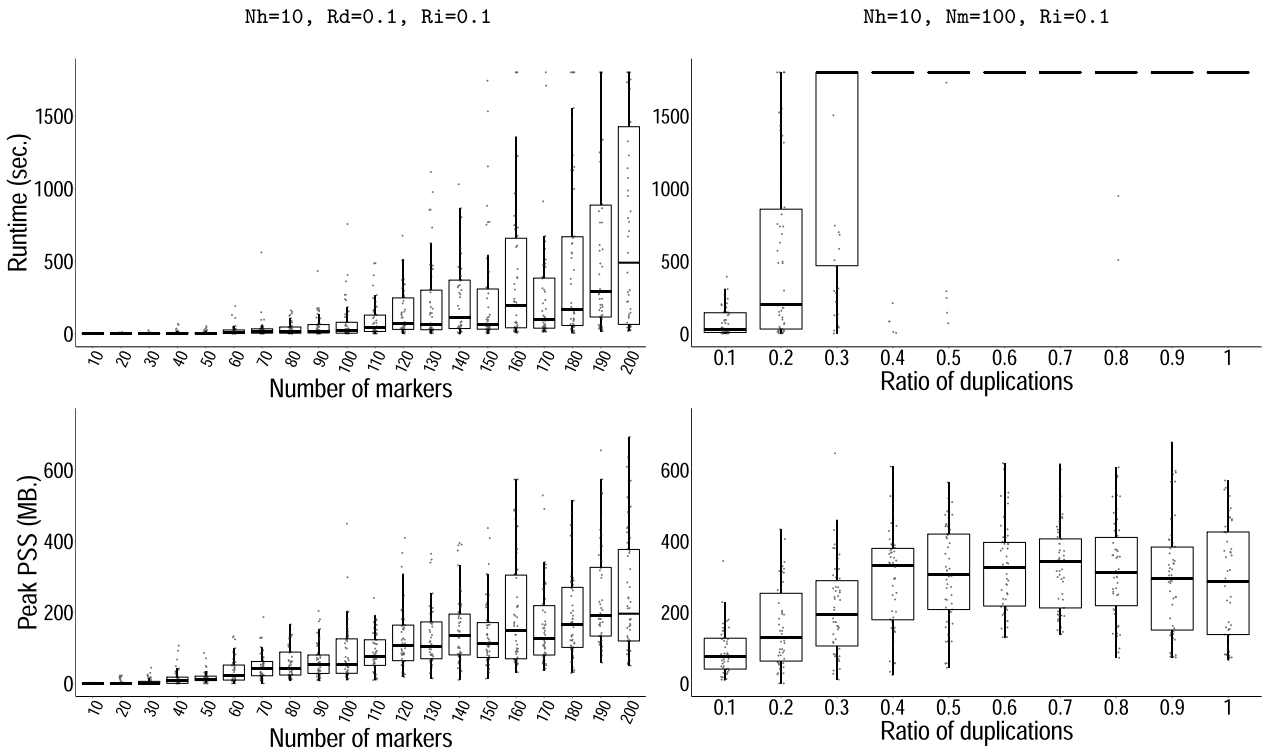


Fig. 11 *Problem 3, recombinations minimization performance benchmarks.* Plots analogous to Fig. 10. Runtime in seconds (upper panels) and peak PSS in MB (lower panels), as a function of the number of markers (left) and of the ratio of duplications (right). For each experiment, the remaining parameters are fixed as indicated above. The abbreviations read as follows: *Nh* number of haplotypes; *Nm*, number of markers; *Rd* ratio of duplications; and *Ri*, ratio of inverted duplications

any role in performance (Additional file 1: Figure S2). For the remaining three, going beyond instances of 200 distinct markers, 20% of duplications, or 40 haplotypes typically did not allow for the optimization to finish in a reasonable amount of time (Fig. 11, Additional file 1: Figure S2). A similar but much less pronounced trend was seen with memory usage, which still remained relatively low. Peak memory usage was again observed at extreme parameter values with a PSS of 1072 MB with 50 haplotypes.

Application: locus 1p36.13

We obtained data from 68 human haplotypes (two per 34 individuals) at the 1p36.13 locus from Porubsky et al. [22] and the T2T-CHM13 human reference sequence [16]. The sequences comprise only eight distinct markers, terminal markers included. The sequences are attributed to five super populations, out of which 18 are of African origin (AFR), 16 of Eastern Asian (EAS), 12 of Admixed American (AMR), 12 of European (EUR), and 10 are South Asian (SAS). Their variation graph is densely connected with 26 edges (Fig. 12). The 68 haplotypes display a high degree of genetic diversity, with haplotype sequences differing in order, orientation, and copy number of the

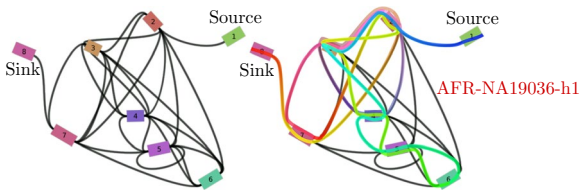


Fig. 12 *Graphical representation of the variation graph for the 1p36.13 locus data.* On the left, a 2D plot rendered by Bandage [43]. Markers are represented as numbered colored rectangles, and the undirected edges connecting them as black curves. Markers 1 and 8 correspond respectively to the source and the sink of the graph. The right plot shows the walk through the graph from source (blue) to sink (red) corresponding to the sequence of haplotype AFR-NA19036-h1, a sample of African origin from our experimental data. The sample's sequence in the previously established notation is: $1\,2\,3\,\overline{4}\,5\,6\,\overline{5}\,\overline{4}\,3\,2\,7\,3\,\overline{2}\,\overline{4}\,3\,\overline{2}\,7\,8$

marker (Suppl. Table T1). Haplotype lengths in terms of the number of markers vary from 15 to 26, with a median of 19.

Our network flow algorithm determined that the data set can be generated from a single founder sequence. Our randomized algorithm for calculation of the minimum number of recombinations in a solution to Problem 1 asserted 15

recombinations after 1M trials, while Algorithm 1 obtained an optimal solution that revealed only 9 recombinations. Minimization completed in 60.3 s with a peak PSS of 225 MB. Note that there exists multiple other co-optimal solutions; Suppl. Figure S4 is an illustration of one.

Discussion

The advent of sequencing technology and genome assembly methodology to reconstruct full human genomes enables research into previously inaccessible segmental duplication loci. This exciting opportunity entails a demand for explanatory models that can infer evolutionary relationships and histories of complex repetitive genomic regions. In this work, we propose a model capable of explaining a broad range of balanced and unbalanced genome rearrangements. Our experiments on simulated data and on the 1p36.13 locus demonstrate that our algorithmic solutions to the founder set problem and the problem of minimizing recombinations in founder sets are capable of processing realistic instances. While the complexity of Problem 3 remains undetermined, we conjecture it to be NP hard.

Importantly, the model we are proposing is based on a molecular mechanism with a well-established role in shaping segmental duplication architecture. In our view, many past models of genome rearrangements have not sufficiently captured biological reality and there is an important need for further research aiming to incorporate knowledge of molecular mechanisms into such models. For instance, we envision future models that additionally include mechanisms like non-homologous end joining (NHEJ) and mobile element insertions. Furthermore, actual rates at which NAHR occurs depend on factors like the length of the duplicated sequence, the sequence similarity, as well as the presence of specific sequence motifs. In our current approach, these aspects are only partially and indirectly captured through the graph construction process. We aim to address and model these factors explicitly in future work.

Abbreviations

DCJ	Double cut and join
HPRC	Human pangenome reference consortium
ILP	Integer linear program
LCA	Least common ancestor
MB	Megabytes
NAHR	Non-allelic homologous recombination
NHEJ	Non-homologous end joining
PSS	Proportional set size
RMQ	Range minimum query
RSS	Resident set size
SD	Segmental duplication
SNP	Single nucleotide polymorphism
SV	Structural variant
USS	Unit set size

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s13015-023-00241-3.

Additional file 1: Figure S1. Reduction in the number of recombinations following minimization. The plots show the total number of recombinations before (blue dots) and after (red dots) minimization, as a function of each simulation parameter. **Figure S2.** Number of recombinations minimization benchmarks. Runtime (upper panels) and peak PSS (lower panels) as a function of the number of haplotypes (left) and the ratio of inverted duplications (right). **Figure S3.** Flow computation performance with a variable ratio of inversions. Runtime (left) and memory usage (right) as a function of this parameter. **Figure S4.** Visualization of a solution to the minimization problem on the 1p36.13 locus. The gray bars correspond to the graph's nodes, labeled 1 to 8. The founder sequence (>1>2>3<7>5>2>3<4>5>5<6<4<3>7<3<2 <4>5>6<5>4<5<4<3 <2>7<3>6>7<3<4<3<2>6<4>3>2>7>8) is traced from top to bottom. A slanted line indicates the underlying node being traversed; if slanted rightwards, traversal is in forward direction, and if slanted leftwards, traversal is in reverse direction. Colors correspond to different haplotypes. The haplotype sequence is: EUR-HG00171-h2, AFR-NA19036-h1, SAS-GM20847-h2, AFR-HG03065-h2, AFR-NA19036-h1, AFR-NA19036-h1, AMR-HG01573-h2, AFR-HG02011-h2, AFR-HG03371-h2, SAS-HG03683-h2. Recombinations are marked with a star. **Figure S5.** Reduction in the number of recombinations following minimization. The plots show the total number of recombinations before (blue dots) and after (red dots) minimization, as a function of each simulation parameter. **Table S1.** Sorted haplotype marker sequences used for analyzing the 1p36.13 locus.

Acknowledgements

The authors kindly thank Dr. Feyza Yilmaz of the Lee Lab (JAX) for providing the haplotype data of the 1p36.13 locus.

Author contributions

TM initiated, TM and DD directed the research project. DD proposed solutions for Problems 1 and 3, and DD and KB implemented the algorithms. DD proposed the suffix tree-based algorithm, KB proposed and implemented the suffix array-based algorithm solving Problem 2. KB developed a method to simulate (N-)AHR, devised and implemented workflows, evaluation tools, and visualizations, and performed the experimental analysis. All authors wrote the manuscript, and read and approved its final version.

Funding

Open Access funding enabled and organized by Projekt DEAL. This work was supported in part by the National Institutes of Health grant 1U01HG010973 to T.M., by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant agreement No 956229, by the BMBF-funded de.NBI Cloud within the German Network for Bioinformatics Infrastructure (de.NBI) (031A532B, 031A533A, 031A533B, 031A534A, 031A535A, 031A537A, 031A537B, 031A537C, 031A537D, 031A538A), and by the MODS project funded from the programme "Profilbildung 2020" (grant no. PROFILNRW-2020-107-A), an initiative of the Ministry of Culture and Science of the State of Northrhine Westphalia.

Availibility of data and materials

All data used for the analysis of the 1p36.13 locus is included in Additional file 1: Table T1. The simulation experiments are solely based on data generated by a program named *hapsim*, available in source code form in the public repository [42].

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 31 March 2023 Accepted: 23 August 2023
Published online: 29 September 2023

References

1. Ukkonen E. Finding Founder Sequences from a Set of Recombinants. In: 2nd International Workshop on algorithms in bioinformatics (WABI 2002). Algorithms in bioinformatics. Berlin, Heidelberg: Springer Berlin Heidelberg; 2002:277-86.
2. Norri T, Cazaux B, Dönges S, Valenzuela D, Mäkinen V. Founder reconstruction enables scalable and seamless pangenomic analysis. Bioinformatics. 2021;37(24):4611–9.
3. Parida L, Melé M, Calafell F, Bertranpetit J, Consortium G. Estimating the ancestral recombinations graph (ARG) as compatible networks of SNP patterns. J Comput Biol. 2008;15(9):1133–53.
4. Swenson KM, Guertin P, Deschênes H, Bergeron A. Reconstructing the modular recombination history of Staphylococcus aureus phages. BMC Bioinform. 2013;14(15):1–9.
5. Rastas P, Ukkonen E. Haplotype Inference Via Hierarchical Genotype Parsing. In: Giancarlo R, Hannenhalli S, editors. 7th International Workshop on Algorithms in Bioinformatics (WABI 2007). Algorithms in Bioinformatics. Berlin, Heidelberg: Springer Berlin Heidelberg; 2007:85-97.
6. Wu Y, Gusfield D. Improved algorithms for inferring the minimum mosaic of a set of recombinants. In: Ma B, Zhang K, editors. Combinatorial Pattern Matching. Springer, Berlin Heidelberg: Berlin, Heidelberg; 2007. p. 150–61.
7. Roli A, Benedettini S, Stützle T, Blum C. Large neighbourhood search algorithms for the founder sequence reconstruction problem. Comput Oper Res. 2012;39(2):213–24.
8. Roli A, Blum C, et al. Tabu search for the founder sequence reconstruction problem: a preliminary study. In: Omatu S, Rocha MP, Bravo J, Fernández F, Corchado E, Bustillo A, et al., editors. Distributed computing, artificial intelligence, bioinformatics, soft computing, and ambient assisted living. Springer, Berlin Heidelberg: Berlin, Heidelberg; 2009. p. 1035–42.
9. Schwartz R, Clark AG, Istrail S. Methods for inferring block-wise ancestral history from haploid sequences. In: 2nd International Workshop on Algorithms in Bioinformatics (WABI 2002). Algorithms in Bioinformatics. Berlin, Heidelberg: Springer Berlin Heidelberg; 2002:44-59.
10. Norri T, Cazaux B, Kosolobov D, Mäkinen V. Minimum Segmentation for Pan-genomic Founder Reconstruction in Linear Time. In: 18th International Workshop on Algorithms in Bioinformatics (WABI 2018). vol. 113 of Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik; 2018:15:1-15:15.
11. Durbin R. Efficient haplotype matching and storage using the positional burrows-wheeler transform (PBWT). Bioinformatics. 2014;30(9):1266–72.
12. Zhao X, Collins RL, Lee WP, Weber AM, Jun Y, Zhu Q, et al. Expectations and blind spots for structural variation detection from long-read assemblies and short-read genome sequencing technologies. Am J Hum Genet. 2021;108:919.
13. Sedlazeck FJ, Lee H, Darby CA, Schatz MC. Piercing the dark matter: bioinformatics of long-range sequencing and mapping. Nat Rev Genet. 2018;19:329.
14. Porubsky D, Ebert P, Audano PA, Vollger MR, Harvey WT, Marijon P, et al. Fully phased human genome assembly without parental data using single-cell strand sequencing and long reads. Nat Biotechnol. 2020;39:302.
15. Ebert P, Audano PA, Zhu Q, Rodriguez-Martin B, Porubsky D, Bonder MJ, et al. Haplotype-resolved diverse human genomes and integrated analysis of structural variation. Science. 2021. https://doi.org/10.1126/science.abf7117.
16. Nurk S, Koren S, Rhie A, Rautiainen M, Bzikadze AV, Mikheenko A, et al. The complete sequence of a human genome. Science. 2022;376(6588):44–53.
17. Wang T, Antonacci-Fulton L, Howe K, Lawson HA, Lucas JK, Phillippy AM, et al. The human pangenome project: a global resource to map genomic diversity. Nature. 2022;604(7906):437–46.
18. Liao WW, Asri M, Ebler J, Doerr D, Haukness M, Hickey G, et al. A Draft Human Pangenome Reference. bioRxiv. 2022. https://www.biorxiv.org/content/early/2022/07/09/2022.07.09.499321.
19. Marques-Bonet T, Girirajan S, Eichler EE. The origins and impact of primate segmental duplications. Trends Genet. 2009;25(10):443–54.
20. Vollger MR, Guitart X, Dishuck PC, Mercuri L, Harvey WT, Gershman A, et al. Segmental duplications and their variation in a complete human genome. Science. 2022;376(6588):eabj6965.
21. Chaisson MJP, Sanders AD, Zhao X, Malhotra A, Porubsky D, Rausch T, et al. Multi-platform discovery of haplotype-resolved structural variation in human genomes. Nat Commun. 2019;10(1):1784.
22. Porubsky D, Höps W, Ashraf H, Hsieh P, Rodriguez-Martin B, Yilmaz F, et al. Recurrent inversion polymorphisms in humans associate with genetic instability and genomic disorders. Cell. 2022;185:1986.
23. Bafna V, Pevzner PA. Genome rearrangements and sorting by reversals. SIAM J Comput. 1996;25(2):272–89.
24. Bader DA, Moret BM, Yan M. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. In: 1st International Workshop on Algorithms in Bioinformatics (WABI 2001). Algorithms in Bioinformatics. Berlin, Heidelberg: Springer Berlin Heidelberg; 2001:365-76.
25. Bafna V, Pevzner PA. Sorting by transpositions. SIAM J Discret Math. 1998;11(2):224–40.
26. Walter MEM, Dias Z, Meidanis J. Reversal and transposition distance of linear chromosomes. In: Proceedings. String Processing and Information Retrieval: A South American Symposium (Cat. No. 98EX207). IEEE; 1998:96-102.
27. Dias Z, Meidanis J. Genome Rearrangements Distance by Fusion, Fission, and Transposition is Easy. In: spire. Citeseer; 2001:250-3.
28. Yancopoulos S, Attie O, Friedberg R. Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics. 2005;21(16):3340–6.
29. Bergeron A, Mixtacki J, Stoye J. A Unifying View of Genome Rearrangements. In: Bucher P, Moret BME, editors. 6th International Workshop on Algorithms in Bioinformatics (WABI 2006). vol. 4175 of Algorithms in Bioinformatics. Berlin, Heidelberg: Springer Berlin Heidelberg; 2006:163-73.
30. Shao M, Lin Y, Moret BME. An exact algorithm to compute the double-cut-and-join distance for genomes with duplicate genes. J Comput Biol. 2015;22(5):425–35.
31. Bohnenkämper L, Braga MD, Doerr D, Stoye J. Computing the rearrangement distance of natural genomes. J Comput Biol. 2021;28(4):410–31.
32. Rautiainen M, Marschall T. MBG: minimizer-based sparse de Bruijn graph construction. Bioinformatics. 2021;37(16):2476–8.
33. Li H, Feng X, Chu C. The design and construction of reference pangenome graphs with minigraph. Genome Biol. 2020;21(1):1–19.
34. Garrison E, Guarracino A, Heumos S, Villani F, Bao Z, Tattini L, et al.. pggb: the PanGenome Graph Builder; 2023. [Paper submission pending; Online, accessed 27-January-2023]. https://github.com/pangenome/pggb.
35. Höps W, Rausch T, Ebert P, Human Genome Structural Variation Consortium (HGSVC), Korbel JO, Sedlazeck FJ. Impact and characterization of serial structural variations across humans and great apes; 2023.
36. Gutin G, Jones M, Sheng B, Wahlström M, Yeo A. Chinese postman problem on edge-colored multigraphs. Discrete Appl Math. 2017;217:196–202.
37. Ahuja RK, Magnanti TL, Orlin JB. Network Flows: Theory, Algorithms, and Applications. 1st ed.; 1993.
38. Gusfield D. Algorithms on stings, trees, and sequences: computer science and computational biology. Acm Sigact News. 1997;28(4):41–60.
39. Ukkonen E. On-line construction of suffix trees. Algorithmica. 1995;14(3):249–60.
40. Matsakis ND, Klock II FS. The rust language. In: ACM SIGAda Ada Letters. vol. 34(3). ACM; 2014. p. 103-4.
41. Gurobi Optimization L. Gurobi Optimizer Reference Manual; 2019. http://www.gurobi.com.
42. Bonnet K, Doerr D. Analysis of the set of founder sequences under a homologous recombination model; 2023. https://github.com/marschall-lab/hrfs.
43. Wick RR, Schultz MB, Zobel J, Holt KE. Bandage: interactive visualization of de novo genome assemblies. Bioinformatics. 2015;31(20):3350–2. https://doi.org/10.1093/bioinformatics/btv383.

Publisher's Note