Genotype Sequence Segmentation: Handling Constraints and Noise

Qi Zhang, Wei Wang, Leonard McMillan, Jan Prins, Fernando Pardo-Manuel de Villena, and David Threadgill

UNC Chapel Hill

Abstract. Recombination plays an important role in shaping the genetic variations present in current-day populations. We consider populations evolved from a small number of founders, where each individual's genomic sequence is composed of segments from the founders. We study the problem of segmenting the genotype sequences into the minimum number of segments attributable to the founder sequences. The minimum segmentation can be used for inferring the relationship among sequences to identify the genetic basis of traits, which is important for disease association studies. We propose two dynamic programming algorithms which can solve the minimum segmentation problem in polynomial time. Our algorithms incorporate biological constraints to greatly reduce the computation, and guarantee that only minimum segmentation solutions with comparable numbers of segments on both haplotypes of the genotype sequence are computed. Our algorithms can also work on noisy data including genotyping errors, point mutations, gene conversions, and missing values.

1 Introduction

Recombination plays an important role in shaping the genetic variations present in current-day populations. Understanding the genetic variations and the genetic basis of traits is crucial for disease association studies. In this paper, we assume an evolution model (previously proposed and studied in [U, WG]) where a population is evolved from a small number of founder sequences. A real-world biological scenario is the Collaborative Cross (CC). The CC [THW, C] is a large panel of 1000 recombinant inbred (RI) mouse strains that were generated from a funnel breeding scheme initiated with a set of 8 founder strains followed by 20 generations of inbreeding. These 8 genetically diverse founder strains capture nearly 90% of the known variations present in the laboratory mouse. The resulting RI strains have a population structure that randomizes the known genetic variation, which will provide unparallel power for disease association studies.

Given a set of founder haplotype sequences, a sequence in the generated population is composed of segments from the founders. It is of interest to identify and label these segments according to their contributing founder. Although the segmentation for a haplotype sequence may be straightforward to compute, in many cases the sequence to be segmented is a genotype sequence for which the two haplotypes are not known completely and they may have different segmentations. For example, the genotype sequence for the strains generated during the intermediate generation in a 20 inbreeding generations of the CC contains two different haplotypes. In this paper, we study the segmentation problem of genotype sequences with the optimization for the minimum number of segments contained in the two associated haplotypes. Furthermore, we extend this basic model to include additional biologically-motivated constraints as well as noise. Since each autosome undergoes, on average, one recombination per meiosis, one expects that the number of founder switches per haplotype at a given generation of breeding to be comparable. Moreover, noise may exist in the founder sequences and the genotype sequence to be segmented. Sources of genotyping noise are both technical and biological. They include point mutations, gene conversions, genotyping errors, etc. Missing genotyping values are also very common in biological data sets.

Similar but different models were studied in [U, WG]. Ukkonen [U] first proposed the founder set reconstruction problem under the assumption that the sample set is evolved from a small set of founders. A dynamic programming algorithm was proposed which computes a minimum number of founders with a given set of sample haplotype sequences, where a segmentation of all the sequences in the sample set can be derived which contains the minimum number of founder switches. Wu and Gusfield [WG] proposed improved polynomial time algorithms for haplotype as well as genotype sample sequences for the special case where there are only two founders. Different from the problems considered in [WG, U], we study the problem where the set of founder sequences are already known, and compute the minimum segmentation for genotype sequences under biologically-relevant constraints and noise. A motivating biological example is the segmentation of the genotype sequences obtained from immediate generations of the CC to estimate the location of the recombination breakpoints. There is other related work on inferring the structure of the variation of the sequences, which include identifying haplotype blocks [DRSHL, GSNM, SHBCI], computing the phylogenies [GEL, G], etc.

In this paper, we propose two dynamic programming algorithms to compute the minimum segmentations for genotype sequences. Our algorithms run in polynomial time and consider biological constraints of the genotype segmentation problem, *i.e.*, the number of segments in both haplotypes are comparable. Moreover, our algorithms account for the potential noise sources in the data including point mutations, gene conversions, genotyping errors, and missing values.

2 The Minimum Segmentation Problem

Assume that we have a set of founding haplotypes $FS = \{F_1, \ldots, F_n, \ldots, F_N\}$. Each haplotype sequence is of length L: $F_n = f_1^n f_l^n \ldots f_L^n$, where $f_l^n \in \{0, 1\}$. Given an input sequence from a population which is derived exclusively from the founder set FS, we are interested in finding a possible segmentation of the sequence, where each segment is inherited from the corresponding region of one of the founders. We first consider the simple case where the input sequence is a haplotype, and then investigate the more interesting case where the input is a genotype sequence.

Given a haplotype sequence, $H = h_1 \ldots h_L$, ($h_l \in \{0, 1\}$), a segment of H is denoted as $\overline{H}_k = h_{s_k} h_{s_k+1} h_{s_k+L_k-1}$, where s_k is the starting position of $\overline{H}_k$, and L_k is the length of $\overline{H}_k$. We consider a segmentation of H which divides the entire sequence into an ordered list of disjoint segments $Seg(H) = \{\overline{H}_1, \ldots, \overline{H}_k, \ldots, \overline{H}_K\}$, where each segment $\overline{H}_k$ is identical to the corresponding region of one of the founders and K is the number of segments in $Seg(H)$. In other words, for each segment $\overline{H}_k = h_{s_k} h_{s_k+1} h_{s_k+L_k-1}$, there exists a founder $F_n = f_1^n f_l^n \ldots f_L^n$ such that $h_{s_k+l_i} = f_{s_k+l_i}^n$, for $l_i = 0, 1, \ldots, L_k - 1$. Furthermore, a minimum segmentation is defined as the segmentation which contains the minimum number of segments. We denote the minimum segmentation as $MinSeg(H) = \{\overline{H}_1, \ldots, \overline{H}_{K_{min}}\}$, where $K_{min} = |MinSeg(H)|$ is the number of segments in $MinSeg(H)$.

If the input is a genotype sequence, we know that it represents two copies of different haplotype sequences, H_a and H_b. Assume that the genotype sequence is $G = g_1 \ldots g_L$, where $g_l \in \{0, 1, 2\}$. A site l is *homozygous* if $g_l = 0$ ($h_l^a = h_l^b = 0$) or $g_l = 1$ ($h_l^a = h_l^b = 1$); a site l is *heterozygous* if H_a and H_b take different alleles, in which case, $g_l = 2$. The process of determining whether $h_l^a = 0, h_l^b = 1$ or $h_l^a = 1, h_l^b = 0$ for a heterozygous site l is called *phasing*. The procedure of determining the two haplotype sequences from the genotype sequence by phasing all the heterozygous sites is called *Haplotype Inference*. For the genotype input case, a segmentation $Seg(G)$ consists of segmentations for both haplotype sequences: $Seg_a(H_a)$ and $Seg_b(H_b)$. The number of segments in $Seg(G)$ is the sum of the numbers of segments in $Seg_a(H_a)$ and $Seg_b(H_b)$: $|Seg(G)| = |Seg_a(H_a)| + |Seg_b(H_b)|$. The minimum segmentation is the segmentation which contains the minimum total number of segments: $|MinSeg(G)| = min\{|Seg(G)|\}$. Let $MinSeg(G) = \{Seg_a^*(H_a), Seg_b^*(H_b)\}$.

In this paper, we develop efficient algorithms for the minimum segmentation problem especially for the genotype input case. In addition to the basic models, there are other issues we may need to consider, such as genotyping errors, point mutations, missing values, the balance of the number of segments in both haplotypes, etc. We will explain later how we model these biological constraints and noise in our solutions.

Solutions for Haplotype Input: Computing the minimum segmentation for the haplotype input sequence is relatively easy and has been discussed in previous studies [WG, U]. A simple greedy algorithm can be applied to compute a minimum segmentation solution by scanning from left to right. Assume that the current site is i (initially it is site 1), and we have a minimum segmentation solution for the part of the input sequence from site 1 to site i. Starting from site i, we try to find the segment shared by the input sequence and one of the founders which extends furthest to the right. This greedy algorithm generates one of the minimum segmentation solutions.

A graph-based dynamic programming algorithm can be used to compute all minimum segmentation solutions given the input haplotype sequence and the founder set. At a high level, we first compute all maximal shared intervals between the input sequence and each founder sequence. The maximal shared interval between the input sequence and founder n is a region where the input sequence is exactly the same as founder n. We consider each shared interval as a node and connect two intervals with an edge if they overlap. In this way, a minimum segmentation solution corresponds to a shortest path from a node starting at the first site to a node ending at the last site. The complete set of the shortest paths can be computed, which are all possible minimum segmentation solutions.

3 Solutions for Genotype Input

The greedy algorithm and the graph-based algorithm for segmenting haplotype input sequences cannot be easily applied on genotype input. The major issue is that we do not know the exact sequences of the two haplotypes due to the multiple possible allele pairs at heterozygous sites. Second, the minimum segmentation solution for the genotype may not consist of the minimum segmentation solutions for each haplotype sequence.

In the following discussion, we describe two dynamic programming algorithms for solving the minimum segmentation problem for genotype input sequences. The first algorithm considers each site separately, the second algorithm considers a region of sites simultaneously, and is thus more efficient.

Site-based Dynamic Programming Algorithm: We consider for each site l, the possible founders for the two haplotype sequences H_a and H_b. If site l is a homozygous site, assuming $g_l = 0$ (without loss of generality), we have $h_l^a = h_l^b = 0$. Let $of^{a,l}$ be the original founder where h_l^a was inherited from at site l. Then $of^{a,l}$ must be one of the founders which also take 0 at site l: $of^{a,l} \in \{F_n | f_l^n = 0\}$. Similarly, we have the founder where h_l^b was inherited from as: $of^{b,l} \in \{F_n | f_l^n = 0\}$. Let $fp^l = \langle of^{a,l}, of^{b,l} \rangle$ denote the possible founder pair at site l, we have the set of all possible founder pairs as $FP^l = \{fp^l | fp^l \in \{F_n | f_l^n = 0\} \times \{F_n | f_l^n = 0\}\}$. If site l is a heterozygous site where $g_l = 2$, there are two possibilities: $h_l^a = 1 \wedge h_l^b = 0$ or $h_l^a = 0 \wedge h_l^b = 1$. Therefore, the possible founder pairs for heterozygous site l is $FP^l = \{fp^l | fp^l \in \{F_n | f_l^n = 0\} \times \{F_n | f_l^n = 1\} \cup \{F_n | f_l^n = 1\} \times \{F_n | f_l^n = 0\}\}$. We compute the founder pair set FP^l for each site l.

Assigning a founder pair from FP^l to each site l generates a segmentation of the input genotype sequence. The number of segments of both haplotypes (of the genotype) are the total number of *founder switches* between founder pairs of every consecutive sites plus 2. Consider two neighboring sites l and $l+1$. If the corresponding founder pairs are $fp_{q_l}^l = \langle of_{q_l}^{a,l}, of_{q_l}^{b,l} \rangle$ $(1 \le q_l \le |FP^l|)$ and $fp_{q_{l+1}}^{l+1} = \langle of_{q_{l+1}}^{a,l}, of_{q_{l+1}}^{b,l} \rangle$ $(1 \le q_{l+1} \le |FP^{l+1}|)$, the number of founder switches between these two founder pairs $FounderSwitch(fp_{q_l}^l, fp_{q_{l+1}}^{l+1})$ can be computed as:

$$FounderSwitch(fp^l_{q_l}, fp^{l+1}_{q_{l+1}}) = \begin{cases} 0 : \text{if } of^{a,l}_{q_l} = of^{a,l+1}_{q_{l+1}} \wedge of^{b,l}_{q_l} = of^{b,l+1}_{q_{l+1}} \\ 1 : \text{if } of^{a,l}_{q_l} = of^{a,l+1}_{q_{l+1}} \wedge of^{b,l}_{q_l} \neq of^{b,l+1}_{q_{l+1}} \text{ or} \\ \quad\quad of^{a,l}_{q_l} \neq of^{a,l+1}_{q_{l+1}} \wedge of^{b,l}_{q_l} = of^{b,l+1}_{q_{l+1}} \\ 2 : \text{if } of^{a,l}_{q_l} \neq of^{a,l+1}_{q_{l+1}} \wedge of^{b,l}_{q_l} \neq of^{b,l+1}_{q_{l+1}} \end{cases} \tag{1}$$

Let $K_{min}(g_1 \dots g_{l-1}|fp^l_{q_l})$ be the minimum number of segments in any segmentation solution over the subsequence $g_1 \dots g_l$ which at site l takes the founder pair $fp^l_{q_l}$. The minimum number of segments over the entire genotype sequence $K_{min}(g_1 \dots g_L)$ can be computed as:

$$K_{min}(g_1 \dots g_L) = min_{fp^L_{q_L} \in FP^L}\{K_{min}(g_1 \dots g_{L-1}|fp^L_{q_L})\} \tag{2}$$

The main recurrence of the dynamic programming algorithm is as follows:

$$\begin{aligned} K_{min}(g_1 \dots g_{l-1}|fp^l_{q_l}) = min_{fp^{l-1}_{q_{l-1}} \in FP^{l-1}}\{&K_{min}(g_1 \dots g_{l-2}|fp^{l-1}_{q_{l-1}}) + \\ &FounderSwitch(fp^{l-1}_{q_{l-1}}, fp^l_{q_l})\} \end{aligned} \tag{3}$$

And initially,

$$K_{min}(\Phi|fp^1_{q_1}) = 2, \forall fp^1_{q_1} \in FP^1 \tag{4}$$

The solutions for this dynamic programming problem can be easily computed by populating a table T of L rows where row l has at most $|FP^l|$ entries. The entry $T(l, q_l)$, $1 \leq q_l \leq |FP^l|$ is filled with $K_{min}(g_1 \dots g_{l-1}|fp^l_{q_l})$ during the computation. Row 1 is initialized according to Eq.(4), and row $i+1$ is computed after row i. During the computation of $T(l, q_l)$ according to Eq.(3), we keep the backtracking pointers from entry $T(l, q_l)$ to any $T(l-1, q_{l-1})$ where the minimum values are obtained. In this way, we are able to obtain all the minimum segmentation solutions.

There are at most N^2 founder pairs for each site l, *i.e.*, $|FP^l| \leq N^2, \forall l$. Therefore, the table we populate is of size $O(LN^2)$. It takes constant time to compute $FounderSwitch(fp^l_{q_l}, fp^{l+1}_{q_{l+1}})$, then filling out a single entry in the table takes $O(N^2)$ time. Therefore, the computational complexity for the entire algorithm is $O(LN^4)$. The space complexity is $O(LN^2)$. For very long sequences and a small number of founders, i.e., $L \gg N^4$, the algorithm has linear time and space complexity in terms of the length of the input sequence. If we keep multiple backtrack pointers for each entry while populating the table, we are able to obtain all the minimum segmentation solutions.

Region-based Dynamic Programming Algorithm: For very long sequences, we propose a more efficient algorithm which considers a subregion of the entire sequence instead of a site at a time.

We first consider the homozygous regions, which are the regions of homozygous sites between any two consecutive heterozygous sites. Within a homozygous region, both copies of the haplotype sequences are the same and we know the exact allele at each site. Fig. 1 illustrates an example of a set of four founders $(F_1 - F_4)$ and a genotype input sequence G to be segmented. The length of

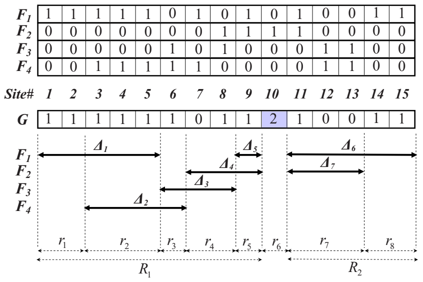


Fig. 1. An example subregions. F_1-F_4 are four founder sequences. G is the genotype sequence to be segmented. There are 15 sites in all sequences, where site 10 is the only heterozygous site. $R_1 : [1, 9]$ and $R_2 : [11, 15]$ are the homozygous regions. Δ_1-Δ_5 are the maximal shared intervals in R_1. Δ_6 and Δ_7 are the maximal shared intervals in R_2. $r_1 - r_8$ are the subregions for the entire sequence, out of which r_6 is the heterozygous subregions, and the remaining are the homozygous subregions.

each founder and the genotype sequence is 15, with 14 homozygous sites and 1 heterozygous site (site 10). The homozygous regions are $R_1 = [1, 9]$ and $R_2 = [11, 15]$. For each homozygous region, we compute all the maximal shared intervals between each founder and the haplotype sequences. A maximal shared interval Δ_i is an interval over which a haplotype and a founder shares the same allele at each site and the region cannot be extended further on either side. We represent each maximum shared interval as a triple, for example, $\Delta_i : (I_i, H_a, F_n)$ is a maximal shared interval between haplotype H_a and founder F_n over interval I_i. Since both haplotypes are the same, a maximal shared interval for haplotype H_a is also a maximal shared interval for haplotype H_b, therefore, the maximal shared interval for the homozygous regions can also be represented as $\Delta_i : (I_i, *, F_n)$. In Fig. 1, $\Delta_1 - \Delta_5$ are the maximal shared intervals within region R_1 for both haplotype sequences. We divide each homozygous region R_j into a set of subregions using the two end points of all maximal shared intervals inside R_j. For example, in Fig. 1, R_1 is divided into subregions r_1, r_2, r_3, r_4, and r_5. If we consider each heterozygous site as a 1-site subregion (e.g. r_6 in Fig. 1), together with all the subregions for the homozygous regions, we have a complete set of subregions $\{r_p\}$ which cover the entire sequence (e.g., $r_1 - r_8$ in Fig. 1).

For each homozygous subregion r_p, let $fp^{r_p} = \langle of^{a,r_p}, of^{b,r_p} \rangle$ be a possible founder pair for subregion r_p. We know that the set of possible founder pairs is $FP^{r_p} = \{\langle of^{a,r_p}, of^{b,r_p} \rangle |\ \exists \Delta_{i_1} = (I_{i_1}, *, of^{a,r_p}), \Delta_{i_2} = (I_{i_2}, *, of^{b,r_p})$, where $I_{i_1} \supseteq r_p, I_{i_2} \supseteq r_p\}$. For example, the founder pair for the subregion r_2 in Fig. 1 could be $\langle F_1, F_1 \rangle$, or $\langle F_1, F_2 \rangle$, or $\langle F_2, F_1 \rangle$, or $\langle F_2, F_2 \rangle$. For each heterozygous subregion which is composed of a heterozygous site l, since h^l_a and h^l_b take different alleles, any possible founder pair should consist of a founder taking allele 1 and a founder taking allele 0. For example, in Fig. 1, the possible founder pairs for r_6 are $\langle F_1, F_2 \rangle$, $\langle F_2, F_1 \rangle$, $\langle F_2, F_3 \rangle$, $\langle F_2, F_3 \rangle$, $\langle F_2, F_4 \rangle$, and $\langle F_4, F_2 \rangle$.

Instead of considering each site, we consider each subregion in the dynamic programming solution. Assign $fp^{r_p}_{q_p} = \langle of^{a,r_p}_{q_p}, of^{b,r_p}_{q_p} \rangle$ to be the founder pair for subregion r_p, where $1 \leq q_p \leq |FP^{r_p}|$, and $fp^{r_{p+1}}_{q_{p+1}} = \langle of^{a,r_p}_{q_{p+1}}, of^{b,r_p}_{q_{p+1}} \rangle$ to be the founder pair for subregion r_{p+1}, where $1 \leq q_{p+1} \leq |FP^{r_{p+1}}|$. Similarly, we count the number of founder switches between $fp^{r_p}_{q_p}, fp^{r_{p+1}}_{q_{p+1}}$ as:

$$FounderSwitch(fp^{r_p}_{q_p}, fp^{r_{p+1}}_{q_{p+1}}) = \begin{cases} 0 : \text{if } of^{a,r_p}_{q_p} = of^{a,r_{p+1}}_{q_{p+1}} \wedge of^{b,r_p}_{q_p} = of^{b,r_{p+1}}_{q_{p+1}} \\ 1 : \text{if } of^{a,r_p}_{q_p} = of^{a,r_{p+1}}_{q_{p+1}} \wedge of^{b,r_p}_{q_p} \neq of^{b,r_{p+1}}_{q_{p+1}} \\ \quad\; of^{a,r_p}_{q_p} \neq of^{a,r_{p+1}}_{q_{p+1}} \wedge of^{b,r_p}_{q_p} = of^{b,r_{p+1}}_{q_{p+1}} \\ 2 : \text{if } of^{a,r_p}_{q_p} \neq of^{a,r_{p+1}}_{q_{p+1}} \wedge of^{b,r_p}_{q_p} \neq of^{b,r_{p+1}}_{q_{p+1}} \end{cases} \tag{5}$$

Let $K_{min}(r_1 \dots r_{p-1}|fp^{r_p}_{q_p})$ be the minimum number of segments in any segmentation solution over the subsequence covered by $r_1 \dots r_p$ which takes the founder pair $fp^{r_p}_{q_p}$ at subregion r_p. The minimum number of segments over the entire genotype sequence $K_{min}(r_1 \dots r_P)$ where r_P is the last subregion can be computed as:

$$K_{min}(r_1 \dots r_P) = min_{fp^{r_P}_{q_P} \in FP^{r_P}} \{K_{min}(r_1 \dots r_{P-1}|fp^{r_P}_{q_P})\} \tag{6}$$

The main recurrence of the dynamic-programming algorithm is as follows:

$$\begin{aligned} K_{min}(r_1 \dots r_{p-1}|fp^{r_p}_{q_p}) = min_{fp^{r_{p-1}}_{q_{p-1}} \in FP^{p-1}} \{ & K_{min}(r_1 \dots r_{p-2}|fp^{r_{p-1}}_{q_{p-1}}) + \\ & FounderSwitch(fp^{r_{p-1}}_{q_{p-1}}, fp^{r_p}_{q_p})\} \end{aligned} \tag{7}$$

And initially,

$$K_{min}(\Phi|fp^{r_1}_{q_1}) = 2, \forall fp^{r_1}_{q_1} \in FP^{r_1} \tag{8}$$

We can also solve this dynamic programming problem by populating a table T which contains P rows where row p has at most $|FP^{r_p}|$ entries. Entry $T(p, q_p), 1 \leq q_p \leq FP^{r_p}$ is filled with $K_{min}(r_1 \dots r_{p-1}|fp^{r_p}_{q_p})$ during the computation. There are at most N^2 founder pairs for each subregion r_p, *i.e.*, $|FP^{r_p}| \leq N^2$. The table we populate is of size $O(PN^2)$. The computation of all the maximal shared intervals is $O(LN)$, and filling out each entry in the table costs $O(N^2)$. Thus the computational complexity of region-based dynamic programming algorithm is $O(LN + PN^4)$. Compared with the site-based algorithm which has a time complexity of $O(LN^4)$, if P is much smaller than L, we can greatly reduce the running time, especially for large L.

3.1 Enforcing the Constraints and Modeling Noise

Comparable Number of Founder Switches on Both Haplotypes: During each meiosis autosomes undergo one recombination on average. Thus, during the development of an recombinant inbred-line (RIL), one expects that the number of founder switches per haplotype at each generation to be comparable.

During each mating in the evolving history, each of the two haplotypes may be generated by a new recombination. Therefore, we may expect that for the given genotype to be segmented, the number of segments for the two haplotype sequences are comparable.

For a segmentation $Seg(G)$ of genotype sequence G, which is composed of a segmentation $Seg_a(H_a)$ on haplotype H_a and a segmentation $Seg_b(H_b)$ on haplotype H_b, we put an extra constraint on the minimum segmentation as follows: the difference of the numbers of the segments in the two haplotypes is no more than a threshold α: $||Seg_a(H_a)|-|Seg_b(H_b)|| < \alpha$, where α is a nonnegative integer. The definition of the minimum segmentation for G is then modified as: $|MinSeg(G)| = min\{|Seg(G)|\}$, where $MinSeg(G) = \{Seg_a^*(H_a), Seg_b^*(H_b)\}$ and $||Seg_a^*(H_a)| - |Seg_b^*(H_b)|| < \alpha$.

To compute the minimum segmentation solution with constraints, we propose an efficient heuristic which prunes out solutions that do not satisfy the constraint before they are fully computed during the table population process. This greatly reduces the computation, especially when there are a lot of minimum segmentation solutions that do not satisfy the constraint. We will explain how it works with the region-based algorithm. Assume that we have a founder set $\{F^1, \ldots, F^N\}$, and a genotype sequence G. For any minimum segmentation solution $MinSeg(G) = \{Seg_a^*(H_a), Seg_b^*(H_b)\}$, we have the following lemma:

Lemma 1. *For any homozygous region R on G, and any minimum segmentation solution $\{Seg_a^*(H_a), Seg_b^*(H_b)\}$, let the set of segments in $Seg_a^*(H_a)$ which completely fall inside R be $Seg_a^*(H_a) \cap R$, and the set of segments in $Seg_b^*(H_b)$ which completely fall inside R be $Seg_b^*(H_b) \cap R$, then we have*

$$||Seg_a^*(H_a) \cap R| - |Seg_b^*(H_b) \cap R|| \leq 2 \tag{9}$$

Proof. Details of the proof are presented in [ZWMPVT].

Lemma 2. *For a genotype sequence G containing Z heterozygous sites, any of its minimum segmentation $\{Seg_a^*(H_a), Seg_b^*(H_b)\}$ satisfies:*

$$||Seg_a^*(H_a)| - |Seg_b^*(H_b)|| \leq 3Z + 1 \tag{10}$$

Proof. For more details, please refer to [ZWMPVT].

We can use Lemma 2 in the dynamic programming algorithm when we are populating the table. Assume currently we are filling out the entry $T(p, q_p)$, *i.e.*, we are computing $K_{min}(r_1 \ldots r_{p-1}|fp_{q_p}^{r_p})$ according to Eq.(7), which is the minimum segmentation for the subsequence from r_1 to r_p with $fp_{q_p}^{r_p}$ as the founder pair for r_p. In addition to computing the minimum segmentation, we also keep track of the difference between the number of segments over two haplotypes in the minimum segmentation we have computed, $\delta(p, q_p) = ||Seg_a^*(H_a[r_1, r_p])| - |Seg_b^*(H_b[r_1, r_p])||$. Let the number of heterozygous sites in the remaining part of the sequence be $Z([r_{p+1}, r_P])$. Then if $\delta(p, q_p) - (3Z([r_{p+1}, r_P]) + 1) > \alpha$, according to Lemma 2, we know that the corresponding solution will not be able to generate a minimum segmentation solution for the entire sequence where the difference between the number of segments on both haplotypes is less than α. If the minimum segmentation solution we consider is from $T(p-1, q_{p-1})$ to $T(p, q_p)$, then if $\delta(p, q_p) - (3Z([r_{p+1}, r_P]) + 1) > \alpha$, we will not add the backtrack pointer from $T(p, q_p)$ to $T(p-1, q_{p-1})$.

Modeling Point Mutation, Genotyping Errors and Gene Conversions: There are both biological and technical resources of noise in genotyping, which include point mutations and gene conversions, and genotyping errors. We consider these potential noise sources in the data in our segmentation algorithms.

A point mutation or genotyping error can be treated as a single site mismatch that falls within a shared interval between a copy of haplotype sequence and a founder sequence. A gene conversion can be treated as a short sequence of mismatches that fall within a shared interval between a copy of haplotype sequence and a founder sequence. In the following, we explain how we model these noise sources in our region-based dynamic programming algorithm. We first consider the point mutation, gene conversion, and genotyping error which happen side a homogeneous range. During the computation of maximal shared intervals between sequence G and each founder F_n over a homozygous region R, assume that we have two maximal intervals Δ_1, Δ_2 between G and F_n which are over the intervals $I_1 = [b_1, e_1]$ and $I_2 = [b_2, e_2]$. We know that I_1 and I_2 are not overlapping or touching. Let I_1 be the interval on the left, *i.e.*, $e_1 < b_2 - 1$. If $e_1 = b_2 - 2$, then there is a single mismatch at site $e_1 + 1$ within the combined region $[b_1, e_2]$. Assume that both I_1 and I_2 are of enough length, then this single mismatch may be a point mutation or genotyping error. Therefore, we create another shared interval Δ_3 between G and F_n over the single-site interval $I_3 = [e_1 + 1, e_1 + 1]$. This interval has a probability of $\beta < 1$ to be a shared interval between G and F_n. β is defined as the probability of a single mismatch inside a shared interval being a point mutation or genotyping error. Similarly, if $e_1 < b_2 - 2$ but $b_2 - 1 - e_1 < gc$, which means the gap between interval I_1 and I_2 is shorter than a maximal possible length gc of a typical gene conversion, then this short sequence of mismatches may be a gene conversion, assuming both I_1 and I_2 are of enough length. We create another shared interval Δ_4 between G and F_n over the short interval $I_4 = [e_1 + 1, b_2 - 1]$. This interval has a probability of $\gamma < 1$ to be a shared interval between G and F_n. γ is defined as the probability of a short sequence of mismatches inside a shared interval being a gene conversion. We can consider the point mutation and genotyping error that happen at a heterozygous site in a similar manner, where we check whether the heterozygous site is a single mismatch falling into a shared interval, *i.e.* the shared interval to the left of the heterozygous site and to the right of the heterozygous site are from the same founder. The maximal shared intervals computed without considering noise are of probability 1. By modeling the noise using intervals with probability less than 1, we can compute the minimum segmentation solution with desired noise tolerance $\theta < 1$. When we compute each entry $T(p, q_p)$ in the table, we keep track of the accumulated probability which is the multiplication of the probabilities of the intervals in the founder pairs on the minimum segmentations solution so far. We only keep the solution with the accumulated probability no less than θ.

Modeling Missing Values: Besides noise and incorrect values, there can also be missing values in the data. Assume that the missing value is in founder F_n, at site l, and the value at the same site on the sequence G is not missing. If l is a homozygous site, we fill out f_l^n using g_l. If l is a heterozygous site, we consider f_l^n

can be either 0 or 1 when we create the founder pairs for this heterozygous site. If the missing value is on G at site l, we consider it to be either 0 or 1, which means this site can be in a maximal shared interval with any of the founder, no matter whether the founder has a missing value at the same site or not. In this way, we can generate the minimum segmentation solution with the smallest possible number of segments. The missing values in both founders and genotype sequences are filled with values in each solution (it may be filled with different values for different minimum segmentation solutions).

4 Experimental Results

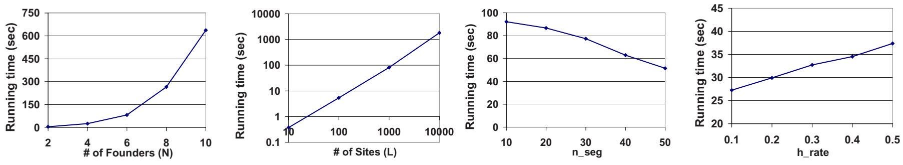
We tested the performance of our region-based dynamic programming algorithm on the simulated data. As we presented earlier, the region-based algorithm has less computational complexity than the site-based algorithm. We only demonstrate the results on region-based algorithm.

The set of the founder sequences $\{F_n\}$ and the genotype sequence G (corresponding two haplotype sequences H_a, H_b) are generated so that: (a) The set of founders are generated randomly, except that, at each site, there is at least a founder taking 0 and at least a founder taking 1, (b) The number of the heterozygous sites in G is $h\_rate \times L$ where h_rate is a parameter representing the occurrence rate of the heterozygous sites, L is the total number of sites, and (c) H_a and H_b are generated by randomly patching up n_seg random segments from the founders.

Note that, the segmentation during the generation provides a lower bound on the number of segments in the minimum segmentation. The computed minimum segmentation may not have the same number of segments on both haplotypes. The code is implemented using Matlab, and the experiments are performed on an Intel Core 2 Duo 1.6GHz machine with 3GB memory.

Running Time. We evaluated the running time by varying the number of founders (N), the number of sites (L), the number of segments (n_seg), and the heterozygous sites occurrence rate (h_rate).

Fig. 2(a) highlights the running time by varying the number of founders from 2 to 10. Other parameters are fixed to be $L = 1000, n\_seg = 30, h\_rate = 0.01$. The complexity of the algorithm is $O(LN + PN^4)$, which is demonstrated by the superlinear increase in the running time with increasing N. Fig. 2(b) shows the running time with varying number of sites. All data sets contain 6 founders. n_seg for 10, 100, 1000, 5000, and 10000-site data sets are chosen to be 3, 8,



(a) Time vs. N (b) Time vs. L (c) Time vs. n_seg (d) Time vs. h_rate

Fig. 2. Running time with varying parameters

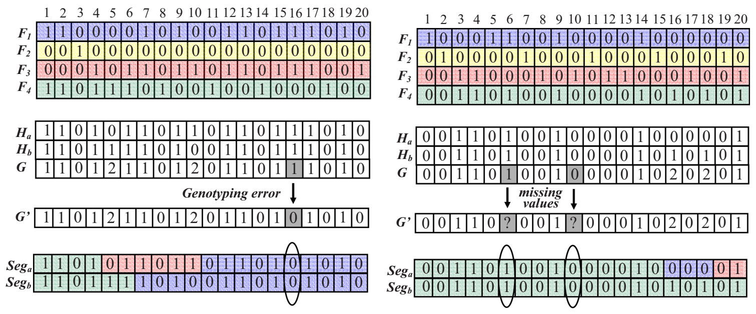

(a) Data with genotyping error (b) Data with missing values

Fig. 3. Segmentation results (better viewed in color)

30, 150, 300, respectively. *h_rate* is 0.01 for all the data sets except for the 10-site set where the *h_rate* is set to be 0.1 to guarantee 1 heterozygous site. Both X-axis and Y-axis are in log scale, we can observe that the running time is linear to the number of sites L. In Fig. 2(c), the running time decreases as *n_seg* increases. The reason is due to the increased number of founder pairs generated from large shared intervals when *n_seg* is small. The data sets have 1000 sites, 6 founders, and *h_rate* is set to 0.01. Fig. 2(d) shows the increase of the running time with increasing *h_rate*. More heterozygous sites introduce more subregions which cause the running time to increase. The data sets contain 1000 sites, 4 founders and *n_seg* is set to 50.

Effect of Enforcing the Constraint. Table 1 shows the results on three datasets where different constraints are applied. As shown in the table, enforcing the constraint greatly reduces the number of minimum segmentation solutions generated. The running time also decreases with the application of the constraints. For dataset #1, there are 28 minimum segmentation solutions where both haplotypes take the same number of segments. However, the number of solutions increases to 40 when the number of segments over the two haplotypes can differ by 1. Similarly, for dataset #2, the number of solutions increased 5 times, and the run time also doubled. A similar trend is observed for dataset #3.

Error Tolerance. We tested our algorithms on simulated data with point mutation, genotyping errors, and missing values. Fig. 3 shows two example segmentation results. In Fig. 3(a), the data set contains four founders $\{F_1, F_2, F_3, F_4\}$ each of which has 20 sites. The two copies of the haplotypes H_a and H_b, and the corresponding genotype G is shown in the figure as the ground truth. A random site chosen to take a genotyping error. The resulting genotype G' has a genotyping error (1 is mistaken as 0) at site 16. Our segmentation solution on G' is shown at the bottom of Fig. 3(a). Although F_1 does not match G at site 16, F_1 is still chosen as the founder in Seg_a and Seg_b, since site 16 is a single mismatch inside a long shared interval with F_1. Fig. 3(b) shows the result on a data set with missing values. Two random sites (site 6 and site 10) are chosen

Table 1. Effect of Enforcing the Constraint

dataset	#1		#2			#3	
parameters	$N = 4, L = 20$ $n\_seg = 4, h = 0.1$		$N = 2, L = 50$ $n\_seg = 8, h = 0.05$			$N = 6, L = 20$ $n\_seg = 6, h = 0.05$	
α	0	1	0	1	2	0	1
# solutions	28	40	92	276	460	6	356
running time (sec)	0.515	0.718	0.453	0.656	0.812	1.09	1.437

to be the sites with missing values. As shown at the bottom of the figure, our algorithm still generates the correct minimum segmentation with the values at both sites filled in. Fig. 3 is better to be viewed in color.

5 Conclusions

In this paper, we studied the minimum segmentation problem for genotype sequence given a set of founders. We proposed dynamic programming algorithms which run in polynomial time. The algorithms can effectively handle the constraint which requires comparable number of founder switches on both haplotypes. Moreover, the algorithms can deal with the noise in the data such as genotyping errors, point mutations, missing values, etc.

References

[C] Churchill, G.A., et al.: The Collaborative Cross, a community resource for the genetic analysis of complex traits. Nat. Genet. 36, 1133–1137 (2002)

[DRSHL] Dally, M., Rioux, J., Schaffner, S.F., Hudson, T., Lander, E.: High-resolution haplotype structure in the human genome. Nat. Genet. 29, 229–232 (2001)

[G] Gusfield, D.: Haplotyping as perfect phylogeny: conceptual framework and efficient solutions. In: RECOMB, pp. 166–175 (2002)

[GEL] Gusfield, D., Eddhu, S., Langley, C.: Optimal, efficient reconstruction of phylogenetic networks with constrained recombination. J. Bioinf. Comput. Biol. 2, 173–213 (2004)

[GSNM] Gabriel, S.B., Schaffner, S.F., Nguyen, H., Moore, et al.: The structure of haplotype blocks in the human genome. Science 296, 2225–2229 (2002)

[SHBCI] Schwartz, R., Halldorson, B.V., Bafna, V., Clark, A.G., Istrail, S.: Robustness of inference of haplotyp block structure. J. Comput. Biol. 10, 13–19 (2003)

[THW] Threadgill, D.W., Hunter, K.W., Williams, R.W.: Genetic dissection of complex and quantitative traits: from fantasy to reality via a community effort. Mamm Genome 13, 175–178 (2002)

[U] Ukkonen, E.: Finding founder sequences from a set of recombinants. In: Guigó, R., Gusfield, D. (eds.) WABI 2002. LNCS, vol. 2452, pp. 277–286. Springer, Heidelberg (2002)

[VFM] Valdar, W., Flint, J., Mott, R.: Simulating the Collaborative Cross: Power of Quantitative Trait Loci Detection and Mapping Resolution in Large Sets of Recombinant Inbred Strains of Mice. Genetics 172(3), 1783–1797 (2006)

[WG] Wu, Y., Gusfield, D.: Improved algorithms for inferring the minimum mosaic of a set of recombinants. In: Ma, B., Zhang, K. (eds.) CPM 2007. LNCS, vol. 4580, pp. 150–161. Springer, Heidelberg (2007)

[ZWMPVT] Zhang, Q., Wang, W., McMillan, L., Prins, J., Villena, F.P., Threadgill, D.: Genotype Sequence Segmentation: Handling Constraints and Noise. UNC Tech. Report (2008)