Inferring the Recent Duplication History of a Gene Cluster

Giltae Song[1], Louxin Zhang[2], Tomáš Vinař[3], and Webb Miller[1]

[1] Center for Comparative Genomics and Bioinformatics, 506B Wartik Lab,
Penn State University, University Park, PA 16802, USA
[2] Department of Mathematics, National University of Singapore, Singapore 117543
[3] Faculty of Mathematics, Physics and Informatics, Comenius University,
Mlynska Dolina, 842 48 Bratislava, Slovakia

Abstract. Much important evolutionary activity occurs in gene clusters, where a copy of a gene may be free to evolve new functions. Computational methods to extract evolutionary information from sequence data for such clusters are currently imperfect, in part because accurate sequence data are often lacking in these genomic regions, making the existing methods difficult to apply. We describe a new method for reconstructing the recent evolutionary history of gene clusters. The method's performance is evaluated on simulated data and on actual human gene clusters.

1 Introduction

Gene clusters are formed by duplication, followed by substitution, inversion, deletion, and/or gene conversion events. The resulting copies of genes provide the raw material for rapid evolution, as redundant copies of a gene are free to adopt new functions [1,2]. A copy may take on a novel, beneficial role that is then preserved by natural selection, a process called *neofunctionalization*, or both copies may become partially compromised by mutations that keep their total function equal to that of the original gene, called *subfunctionalization* [3]. Another source of interest in gene clusters is that several human genetic diseases are caused by a tendency for regions between two copies to be deleted [4]. A major finding of the initial sequencing of the human genome is that 5% of the sequence lies in recent duplications [5]. More recently, it has become clear that duplicated regions often vary in copy number between individual humans [6]. A substantial fraction of what distinguishes humans from other primates, as well as the genetic differences among humans, cannot be understood until we have a clearer picture of the contents of gene clusters and of the evolutionary mechanisms that created them.

One impediment to this understanding is that recently duplicated regions, say those that retain over 95% identity (roughly, that duplicated in the last 10 million years) resist assembly by the current whole-genome shotgun approach [7]. Even the so-called "finished" human genome sequence has 300 gaps, most of which are caused by the presence of recent duplications. Moreover, much

F.D. Ciccarelli and I. Miklós (Eds.): RECOMB-CG 2009, LNBI 5817, pp. 111–125, 2009.

available mammalian genomic sequence is only lightly sampled, and hence even further from supporting analysis of gene clusters. Partly because of the lack of accurate sequence data in gene clusters, practical computational tools for their analysis still need to be developed.

We think of the analysis problem for gene clusters as requiring a marriage of two somewhat distinct approaches, one dealing with large-scale evolutionary operations (primarily duplication, inversion, segmental deletion, and gene conversion) and the other with fine-scale evolution (substitutions and very small insertions/deletions). Even the second part, though it is essentially just an extension of the familiar problem of multiple sequence alignment, is currently not handled well by existing tools. Indeed, just defining what is meant by a proper alignment of a gene cluster sequence is a matter of discussion (e.g., see [8,9]). Although a comparison of several multi-genome alignment programs found reasonable accuracy in single-copy portions of the genome [10], we have found their performance on gene clusters to be inadequate [11]. Our observations about the quality of current whole-genome alignments (e.g., those described in [12]) indicate that it may be worthwhile to align gene clusters using methods designed specifically for them, and then splice the results into the whole-genome alignments created by the other methods.

A number of ideas have been explored for reconstructing large-scale evolutionary history (as opposed to the sequence alignment problem, which deals with substitutions and small insertions/deletions). Some of these attempt to reconstruct the history of duplication operations on regions with highly regular boundaries (e.g., [13,14,15]), some allow inversion (e.g. [16,17]), and also deletions (e.g. [18,19,20]). Typically, whenever more than one group has studied a given formulation of the problem, the methods developed have been fundamentally different, often with large differences in the resulting evolutionary reconstructions and the computational efficiency of the methods.

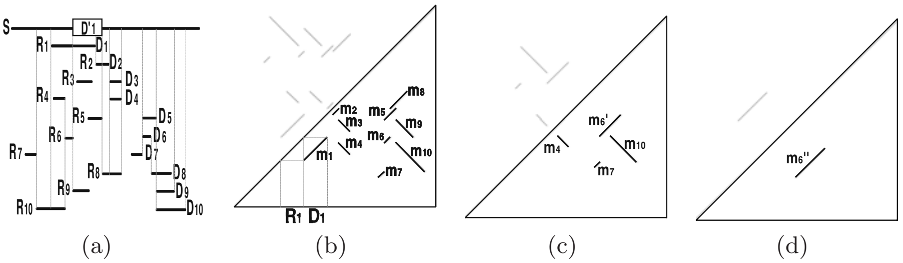
In terms of formulation of the underlying problem, this paper is closest to [18] and particularly [19]. However, the methods described here are quite different. For instance, [19] uses probabilistic techniques whereas our approach is entirely combinatorial.

2 Methods

2.1 Problem Statement and a Basic Algorithm

During genome evolution, a duplication event copies a segment to a new genomic position. Genome sequencing and analysis suggest that a large number of gene clusters in human and other mammalian genomes have been formed by duplication events, often very recent ones. We aim to identify the duplication events that formed a given gene cluster by using a parsimony approach.

Formally, a duplication is an operation that copies a subsequence or its reverse complement into a new position. The original segment is called the *source region* and the inserted copy after the duplication is called the *target region*. Subsequently, the two regions evolve independently by point mutations and small



(a) (b) (c) (d)

Fig. 1. An example of duplication inference. (a) The original matches in a sequence. (b) The self-alignment of the original sequence. (c) The self-alignment after rolling back a duplication. (d) The self-alignment after rolling back two duplications. See Section 2.3 for details.

insertions and deletions and thus in the present day sequence, the two regions are generally not identical. However, they do form a strong local alignment (called a *match*) in a self-alignment of any genomic region that contains them both.

We detect matches by aligning the genomic sequence with itself using the program BLASTZ [21]. Let S be a genomic sequence containing a gene cluster. First, we run BLASTZ to obtain all local self-alignments of S, which are visualized as a dot plot (e.g. Figure 1(b)). These are then processed in a pipeline that links local alignments separated by small gaps and/or interspersed repeats and adjusts their endpoints to avoid inferring tiny spurious duplications. Finally, we identify a set of matches. The problem of inferring the duplication history of the gene cluster is then to find a duplication-free sequence T and the minimum number of duplication events $O_1, O_2, \cdots, O_k$ such that

(i) The source and target regions of each O_i consist of one or more match regions.

(ii) $S = O_k(O_{k-1}(\ldots(O_1(T))))$, where $O_i(S')$ denotes the resulting sequence after applying O to sequence S'.

Given a sequence of duplications $O_1, O_2, \ldots, O_k$, we call the boundaries of all source and target regions *breakpoints*. If duplication events occur randomly during genome evolution, the two duplications are quite unlikely to share their boundaries. So we assume that no two duplications have a common breakpoint in a duplication history [22], except for tandem duplication. Tandem duplication (with or without reversal) copies a source region into a location adjacent to its boundary. Thus, it is a special exception to the *breakpoint uniqueness* assumption.

The algorithm focuses on identifying the latest duplication event in the duplication history of a gene cluster. Once this is inferred, its target region is eliminated. These steps are repeated until the duplication-free sequence is reconstructed.

Before determining the latest target region, we must identify matches that have been split by a subsequent duplication. Consider a given match (R, R'). If a duplication event inserts a segment A in the region R', then the match (R, R') is split into two small matches (R_1, R'_1) and (R_2, R'_2). If this happens, we can correctly identify the original duplication event forming the match (R, R') only if we first identify the one inserting A. Hence, using a kd-tree data structure, we identify all the pairs of matches (R_1, R'_1) and (R_2, R'_2) such that R_1 and R_2 are adjacent but R'_1 and R'_2 are separated by a region A of some match. To guarantee that (R_1, R'_1) and (R_2, R'_2) are not examined before removing A from the sequence, we place $[R_1, A]$, $[R'_1, A]$, $[R_2, A]$, $[R'_2, A]$ into a *suspend list*. We call the regions R_1, R'_1, R_2, R'_2 the *suspended regions*, and the region A on which they depend the *inserted region*.

Definition 1. *We say a region A is contained in a region B if all bases of A are in B but not vice-versa. We say A and B overlap if they share at least one base but neither contains the other.*

Theorem 1. *Assume a sequence S is transformed from a duplication-free sequence T by a series of duplication events. Then the target region of the latest duplication event does not overlap with the source or target regions of any other duplications, and it is not contained in any match regions of S.*

The proof of Theorem 1 follows from the breakpoint uniqueness assumption. Based on Theorem 1, we determine the latest duplication event in the history of a gene cluster as follows. Suppose there are n matches in genomic sequence S. We define the *constraint graph* $G = (V, E)$ of these matches as follows. G is directed and has $2n$ nodes representing the $2n$ regions of the matches. There are three types of arcs. Let (R, R') be a match. If R overlaps a region B of another match, there is an arc from node R to node R'. Such an arc is called a type-1 arc. If R is contained in another match region C, there is an arc from node R to node C, called a type-2 arc. Finally, if $[R, A]$ is in the suspend list, there is an arc from node R to node A, called a type-3 arc. The constraint graph for Figure 1 is given in Figure 3.

By Theorem 1, there must be at least one node with out-degree 0 in a constraint graph. In each loop of the algorithm, we select a node v with out-degree 0 and remove the region corresponding to v from S. If there are several nodes of out-degree 0, the one with the highest similarity level in the self-alignment is selected as the latest duplicated region. By Theorem 2 below, the following algorithm identifies the true number of duplication events and a plausible sequence of such events in $O(n^2 \log n)$.

```
procedure INFER-DUPS($\mathcal{S}$)
Input: A set of matches $\mathcal{S}$ in a self-alignment
Output: A set of duplication events
repeat
   for all the pairs of matches $(R_1, R'_1)$ and $(R_2, R'_2)$ do
      if $R_1$ and $R_2$ are adjacent but $R'_1$ and $R'_2$ are separated by a region $A$
      of some match then
         place $[R_1, A]$, $[R'_1, A]$, $[R_2, A]$, and $[R'_2, A]$ into the suspend list.
      end if
    end for
    $G \leftarrow$ CONSTRUCT-CONSTRAINT-GRAPH ($\mathcal{S}$)
    Identify the regions of out-degree 0 in $G$, and remove the one with the
    highest similarity value from $\mathcal{S}$.
    if the removed region is an inserted region in the suspend list then
      merge the corresponding suspended regions in $\mathcal{S}$.
    end if
    $\mathcal{S} \leftarrow \mathcal{S} - M$, where $M$ is the set of matches that disappear with removal
      of the region
  until $\mathcal{S} = \phi$
  end procedure

  function CONSTRUCT-CONSTRAINT-GRAPH ($\mathcal{S}$)
  for all the pairs of matches $(A, A')$ and $(B, B')$ do
    if $A$ overlaps $B$ then
      type-1 arc of $A \rightarrow A'$ and $B \rightarrow B'$
    else if $A$ is contained in $B$ (or vice versa) then
      type-2 arc of $A \rightarrow B$ (or $B \rightarrow A$)
    else if $A$ (or $B$) is a suspended region with $C$ in the suspend list then
      type-3 arc of $A \rightarrow C$ (or $B \rightarrow C$)
    end if
  end for
  end function
```

Theorem 2. *Suppose a sequence S evolves from a duplication-free sequence T in k duplications. If the breakpoint uniqueness assumption holds, the algorithm identifies a series of k duplications and a duplication-free sequence T' such that T' transforms to S by the identified k duplications.*

2.2 Handling Tandem Duplication

Our model assumption of breakpoint uniqueness may be violated by tandem duplication. Copy-and-paste transposons are an example of frequent reuse of duplication breakpoints. To infer duplication history more accurately, a way of handling this tandem duplication is required. Suppose we have a tandem duplication that copies A into a location adjacent to its boundary. It produces a match (A, A') where A' is adjacent to A. If the copied location of A' is not involved in any other matches, the tandem duplication does not affect the algorithm. But if A' split other matches, the target region is contained in the split matches, which violates Theorem 1. For instance, let m be a match, where $m = (BAD, B''A''D'')$. After the tandem duplication in A occurs, m is split into two matches m_1 and m_2, where $m_1 = (BA, B''A'')$ and $m_2 = (A'D, A''D'')$ (see Figure 2). This causes the algorithm to fail to detect the target region A' because A' is contained in both regions of m_1 and m_2. Fortunately, we observe a property that one region of m_1 has a boundary adjacent to a region of m_2 while

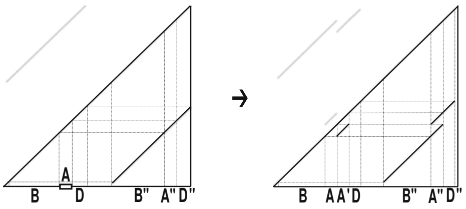


Fig. 2. An example of the self-alignment change caused by a tandem duplication event

the other region of m_1 overlaps the other region of m_2. Also, the boundaries of the overlapped part of m_1 and m_2 correspond to region A''. Thus, the tandem duplication can be detected as follows. If two matches (R_1, R_1') and (R_2, R_2'), where R_1 and R_2 are adjacent but R_1' and R_2' are overlapped (or vice versa), the part where R_1' overlaps R_2' is denoted as a temporary match (A, A'') such that A'' is the overlapped region of R_1' and R_2'. If there exists a match of (A, A') where A and A' are adjacent, then $[R_1, A]$, $[R_1', A]$, $[R_2, A]$, $[R_2', A]$, $[R_1, A']$, $[R_1', A']$, $[R_2, A']$, and $[R_2', A']$ are all placed in the suspend list. In addition, while constructing the constraint graph, if A is contained in a suspended region whose inserted region is A, the drawing of a type-2 arc from A is skipped if A forms a match with the adjacent region.

One potential problem is whether the case we detect as tandem duplication can be generated by other scenarios. Suppose we have three matches m_1, m_2, and m_3 where $m_1 = (BA, B''A'')$, $m_2 = (A'D, A''D'')$, $m_3 = (A, A')$, and that A and A' are adjacent. To simplify, we assume that the out-degree of m_1, m_2, and m_3 in the constraint graph constructed with the other matches is 0. If we consider only parsimonious solutions for this case, the only other scenario is separate duplications of both m_1 and m_2. But m_1 and m_2 cause the violation of the breakpoint uniqueness assumption. If tandem duplication is regarded as a special case of breakpoint reuse, inferring a tandem duplication of m_3 and a duplication of a merged match of m_1 and m_2 makes more sense.

This modification can be extended for tandem duplications which copy a segment more than once into its adjacent location. The tandem duplications of more than one copy are detected as follows. Assume the same source region is involved in all of the copies. If there are $n(\geq 2)$ matches such that $m_1 = (A_1, A_{n+1}), m_2 = (A_1A_2, A_nA_{n+1}), \ldots, m_n = (A_1 \ldots A_n, A_2 \ldots A_{n+1})$, then $m_i (2 \leq i \leq n)$ is converted into m_i' where $m_i' = (A_i, A_{n+1})$. Then, the latest region for this iteration can be identified by running the rest of the algorithm normally.

There is another event that may be confused with tandem duplication. This is a duplication that copies a source region into a location within itself. To handle these correctly, we replace the two matches formed by this type of duplication with one match. This step is motivated by the following. Let m_1 and m_2 be two matches, and suppose we observe $AA'B'B$ or $A\overline{B'A'}B$, where $m_1 = (A, A')$, $m_2 = (B', B)$ and $\overline{B'A'}$ is the reverse complement of sequence $A'B'$. $AA'B'B$ might arise from two duplication events: an event duplicating A and another duplicating B. It could also arise from a single duplication that copies AB within itself. The two-event explanation violates the breakpoint uniqueness hypothesis, and is also less parsimonious than a single event. Therefore, our algorithm infers that $AA'B'B$ arose from a single event that copied and inserted AB within itself. In the same manner, $A\overline{B'A'}B$ also arose from a single event that copied and inserted AB within itself in the reverse orientation. In order to infer one event for two matches in $AA'B'B$, the two matches m_1 and m_2 are replaced with a new match $m' = (AB, A'B')$. The two matches in $A\overline{B'A'}B$ are also replaced in the same way. We call this type of duplication *intraposed duplication*.

2.3 Illustration of the Method

To demonstrate how the method works, we consider a genomic sequence S containing 10 matches m_i, $1 \leq i \leq 10$. The dot plot of the self-alignment of S is shown in Figure 1(b).

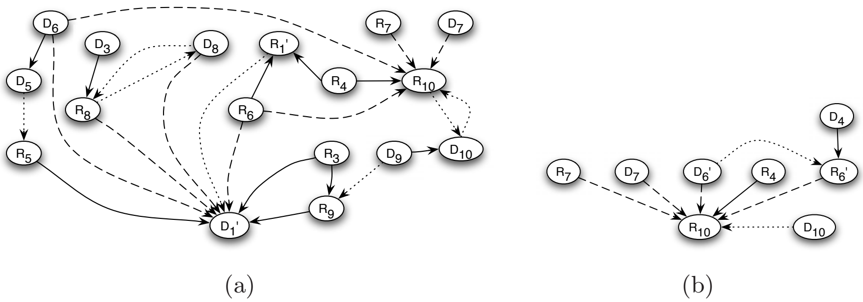
First, we observe that the regions of m_1 and m_2 form a segment $AA'B'B$, where $m_1 = (A, A')$ and $m_2 = (B', B)$, so we infer that they were formed by an intraposed duplication event that inserted a copy of segment AB within itself. We replace m_1 and m_2 with a new match m'_1, whose regions are $R'_1 = AB$ and $D'_1 = A'B'$ respectively. Furthermore, the following two facts are true. Let $m_i = (R_i, D_i)$ for $3 \leq i \leq 10$.

- D_6 and D_8 are adjacent, while R_6 and R_8 are separated by D'_1. Hence, $[R_6, D'_1], [D_6, D'_1], [R_8, D'_1], [D_8, D'_1]$ are added to the suspend list.
- D_6 and D_7 are adjacent, while R_6 and R_7 are separated by R_{10}. Hence, $[R_6, R_{10}], [D_6, R_{10}], [R_7, R_{10}], [D_7, R_{10}]$ are added to the suspend list.

The constraint graph G for the 9 matches is shown in Figure 3(a). Note that there are no arcs leaving node D'_1, so m'_1 is selected as the latest duplication event. After D'_1 is removed from the sequence S, m_6 and m_8 are merged into a match m'_6 in the resulting sequence S'. In addition, since R_3, R_5 and R_9 are contained in D'_1, the matches m_3, m_5 and m_9 do not exist in S'. Overall, in the self-alignment of S' shown in Figure 1(c), only four matches remain, which are m_4, m'_6, m_7, m_{10}. The constraint graph for these four matches is shown in Figure 3(b). Since there are no arcs leaving node R_{10}, we select m_{10} as the latest duplication event. After removal of R_{10}, m'_6 and m_7 are merged into a match m''_6 and m_4 disappears in the resulting sequence S''. As a result, only m''_6 remains in the self-alignment of S''. In summary, we identify 3 duplication events that give rise to the matches in the given genomic sequence S.

2.4 Influence of Deletion and Inversion Events

Deletion events can affect the inference of duplications, so it is important to consider them simultaneously. In order to infer deletions, we use the following procedure. Assume an input sequence S has two segments ABC and $A'C'$ for some non-empty segments A, B, C, A', and C'. We may infer two duplication events that copy A and C respectively, or one duplication that copies ABC and one hypothetical deletion event that deletes B'. Since our goal is to find



(a)

(b)

Fig. 3. The constraint graphs of matches in Fig. 1(b) and 1(c). Arcs of type 1, 2, and 3 are represented by dotted, solid, and dashed lines, respectively. In (a), matches m_1 and m_2 have been replaced with m_1' according to the procedure for intraposed duplications discussed in Section 2.2, node D_4 is omitted because it is identical to D_3. D_4 reappears in (b) after m_3 has been eliminated.

a parsimonious duplication history for the cluster, we infer a duplication event and a deletion event from segments ABC and $A'C'$ when B is relatively short compared to the length of A and C. In our implementation, we detect all possible deletion events by using k-d tree data structure before entering each loop of inferring duplication events.

In the case of inversions, if the inversion does not split any matches generated by duplication events (i.e., it contains the whole region of one or more other matches, or occurs in a region that does not have any matches), then it does not affect the inference of duplication events. Moreover, if the inversion occurs within a match, it can be detected. If it splits other matches by involving source or target regions of duplication events, two duplications are inferred rather than one, but the duplicated regions can be removed correctly.

3 Results

3.1 Human Gene Clusters

We first applied our method to 25 gene clusters in the human genome. For a genomic region containing each gene cluster, we constructed its self-alignment and identified matches using five different thresholds of similarity level: 98%, 93%, 89%, 85%, and 80%, in the same way as [19]. These five similarity levels correspond roughly to the sequence divergence between humans and great apes (GA), old-world monkeys (OWM), new-world monkeys (NWM), lemurs and galagos (LG), and dogs (DOG), respectively. With the matches at different similarity levels, we inferred duplication and deletion events that occurred in different periods in the human lineage. The inferred numbers of duplication and deletion events in these periods are summarized in Table 1.

The human leukocyte antigen (HLA) gene cluster is known to be involved in narcolepsy [23] and celiac disease [24]. In addition, HLA has been observed in

Table 1. Inferred numbers of large-scale duplications and deletions in human gene clusters following divergence from various mammalian clades. GA = great apes (at least 98%); OWM = old world monkeys (93%); NWM = new world monkeys (89%); LG = lemurs and galagos (85%); DOG = dogs (80%). Cluster locations are indicated as coordinates in the March 2006 human genome sequence assembly.

Name	Location	GA	OWM	NWM	LG	DOG
CYP4	chr1:47048227-47411959	1, 0	4, 1	4, 1	5, 1	10, 1
LCE	chr1:150776235-151067237	0, 0	0, 0	5, 1	6, 1	9, 2
CR, DAP3	chr1:153784948-154023311	0, 0	3, 2	12, 2	19, 5	21, 7
FC	chr1:159742726-159915333	1, 2	1, 2	3, 2	4, 2	4, 2
CR1	chr1:205701588-205958677	1, 0	7, 0	8, 3	8, 3	10, 3
CCDC, CFC1	chr2:130461934-131153411	3, 0	5, 0	7, 5	7, 5	10, 5
CXCL, IL8	chr4:74781081-75209572	0, 0	0, 0	0, 0	2, 1	17, 8
PCDH	chr5:140145736-140851366	0, 0	0, 0	0, 0	1, 0	36, 0
HLA	chr6:29786467-30568761	0, 0	2, 0	27, 3	38, 5	50, 6
HLA-D	chr6:33082752-33265289	0, 0	0, 0	0, 0	4, 3	6, 8
OR2	chr7:143005241-143760083	6, 1	8, 1	8, 1	10, 1	16, 1
AKR1C	chr10:4907977-5322660	0, 0	3, 2	4, 3	9, 4	28, 4
GAD2	chr10:26458036-27007198	0, 0	4, 0	6, 1	15, 2	17, 2
PNLIP	chr10:118205218-118387999	0, 0	0, 0	1, 0	2, 1	5, 2
OR5, HB, TRIM	chr11:4124149-6177952	6, 0	7, 0	9, 0	9, 0	24, 2
LST3, SLCO1B	chr12:20846959-21313050	0, 0	0, 0	0, 0	9, 2	21, 3
C14orf	chr14:23177922-23591420	1, 0	4, 0	5, 2	6, 2	8, 3
CYP1A1	chr15:71687352-74071019	2, 0	12, 2	21, 3	23, 4	25, 4
ACSM	chr16:20234773-20711192	2, 0	4, 2	4, 2	4, 2	5, 2
LGALS9, NOS2A	chr17:22979762-23370074	0, 0	2, 0	2, 2	2, 2	3, 2
OR, ZNF	chr19:8569586-9765797	4, 0	6, 0	14, 0	23, 0	33, 0
NPHS1, ZNF	chr19:40976726-43450858	0, 0	6, 1	12, 1	16, 2	23, 4
CYP2A	chr19:46016475-46404199	0, 0	5, 0	11, 3	14, 3	16, 3
DGCR6L, ZNF74	chr22:18594272-19312230	3, 0	4, 0	5, 1	5, 1	5, 1
SLC5A, YWHAH	chr22:30379202-31096691	0, 0	3, 1	4, 1	6, 1	7, 1

the association with prostate cancer [25] and breast cancer [26]. For the HLA gene cluster, the MCMC method of [19] estimated 15 duplications in the lineage between NWM and LG, but our method inferred only 11 duplications, 4 fewer events than the MCMC method (The numbers of the inferred events in the lineage are highlighted in bold in Figure 4(a) and Figure 4(b)).

The aldo-keto reductase (AKR) 1C gene cluster is involved in steroid hormone and nuclear receptors and associated with prostate disease, endometrial cancer, and mammary carcinoma [27]. For this gene cluster, Figure 4(c) and Figure 4(d) show the inference results; our method identified 3 duplications between GA and OWM while the MCMC inferred 5 duplications.

Another interesting observation is that several gene clusters were probably formed by recent gene duplications. For instance, we examined three Cytochrome P450 (CYP) gene clusters, which are associated with lung cancer [28] and esophageal cancer [29]. About 65% of the duplication events inferred for these

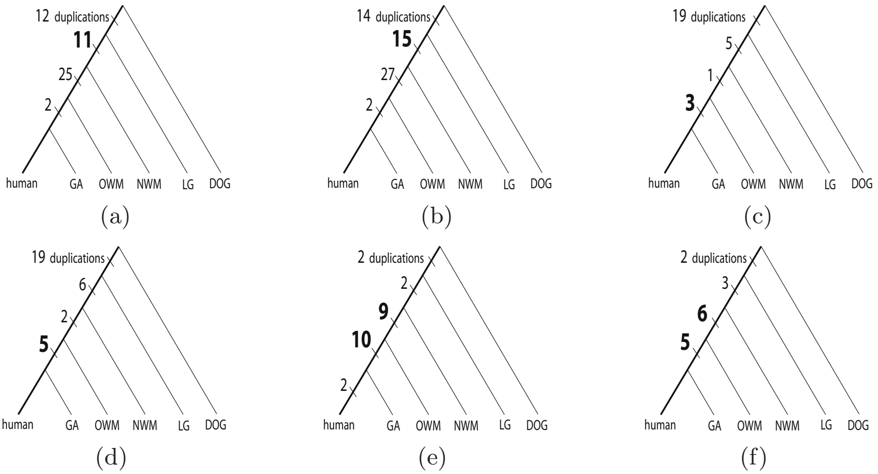


Fig. 4. Duplication events inferred (a) for the HLA gene cluster by the deterministic method, (b) for the HLA gene cluster by the MCMC method, (c) for the AKR1C gene cluster by the deterministic method, (d) for the AKR1C gene cluster by the MCMC method, and (e) for the CYP1A1 gene cluster and (f) the CYP2A gene cluster by the deterministic method

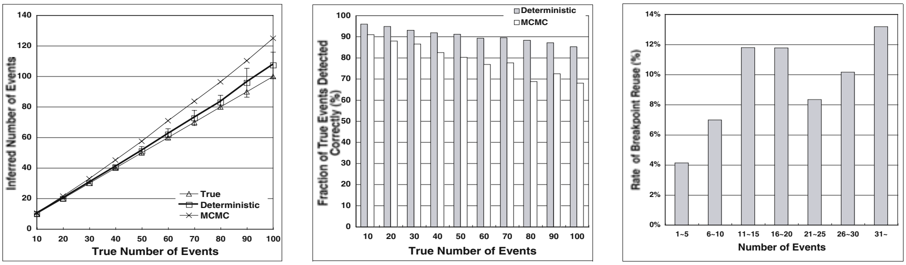
clusters occurred in the evolutionary period between the divergence of humans from new-world monkeys and from great apes. Duplication events inferred for CYP1A1 and CYP2A are mapped onto the phylogeny in Figure 4(e) and Figure 4(f), respectively.

3.2 Validation Test on Simulated Data

Starting from a 500-kb duplication-free sequence, we generated gene clusters by applying a series of duplication events based on the length and distance distributions for duplication and deletion events that we observed in the human genome. We generated 50 gene clusters formed from n duplications for each $n = 10, 20, \ldots, 100$.

On these clusters, our method outperformed the MCMC method reported in [19] in terms of both the total number of inferred duplication events and the number of true duplications detected correctly as indicated in Figure 5(a) and Figure 5(b). On average, our method estimated only 3% events more than true events. A duplication event is expressed as a 3-tuple consisting of a source interval, a target location, and an orientation. If these values for an inferred event exactly match one of the true events, the event is defined to be *correctly detected*, i.e. a true event. We count how many of the inferred events were correctly detected. The fraction of true events detected correctly by our method (91% on average) is much higher than the MCMC method (80% on average).

It is worth noting that duplication breakpoints can be reused in the simulation dataset, since it is generated according to the observed distributions (Figure 5(c))



(a) (b) (c)

Fig. 5. (a) and (b) show the simulation results to evaluate detection of duplication; (a) numbers of reconstructed events and (b) fraction of true events detected correctly. (c) is the rate of breakpoint reuse by the inferred duplications in the human gene clusters.

without constraining the breakpoints to avoid reuse. However, the inferred events are still very close to true events.

4 Discussion and Conclusion

We have developed a combinatorial algorithm that reconstructs recent duplication and deletion operations in a gene cluster from a single present-day sequence. We have compared our combinatorial method with a probabilistic method for the same problem [19], and shown that the relative performance of the combinatorial algorithm is quite good. In addition, a simulation study has validated that our method is very effective for identifying the duplication history.

We are exploring several extensions of this method. The results should be cross-checked against other primate genomes; another, more ambitious goal is to identify the orthology relationship among genes in each gene cluster in the species. Gene conversion should be also considered for more accurate orthology detection rather than using only the overall similarity in the alignment data.

Our goal is to find methods for analyzing large-scale evolutionary operations that integrate well with the specific needs of our current approach for producing whole-genome alignments [12]. We are still in an exploratory stage where the aim is to investigate as many promising avenues as possible. This paper describes a new method whose accuracy, computational efficiency, and focus on an individual species make it a particularly strong contender.

Acknowledgement. GS and WM are supported by Grant HG02238 from NHGRI and LX Zhang is supported by NUS ARF grant R-146-000-109-112. The authors thank Yu Zhang, Cathy Riemer, and Bob Harris for helpful discussions.

References

1. Ohno, S.: Evolution by Gene Duplication. Springer, Berlin (1970)
2. Lynch, M., Conery, J.S.: The evolutionary fate and consequences of duplicate genes. Science 290, 1151–1155 (2000)

3. Force, A., et al.: Preservation of duplicate genes by complementary, degenerative mutations. Genetics 151, 1531–1545 (1999)
4. Lupski, J.R.: Genomic rearrangements and sporadic disease. Nat. Genet. 39(suppl. 7), 43–47 (2007)
5. Lander, E.S., et al.: Initial sequencing and analysis of the human genome. Nature 409(6822), 860–921 (2001)
6. Wong, K.K., et al.: A comprehensive analysis of common copy-number variations in the human genome. Am. J. Hum. Genet. 80(1), 91–104 (2007)
7. Green, E.D.: Strategies for the systematic sequencing of complex genomes. Nat. Rev. Genet. 2(8), 573 (2001)
8. Blanchette, M., et al.: Aligning multiple genomic sequences with the threaded blockset aligner. Genome Res. 14(4), 708–715 (2004)
9. Raphael, B., et al.: A novel method for multiple alignment of sequences with repeated and shuffled elements. Genome Res. 14(11), 2336 (2004)
10. Margulies, E., et al.: Analyses of deep mammalian sequence alignments and constraint predictions for 1% of the human genome. Genome Res. 17(6), 760–764 (2007)
11. Hou, M.: unpublished data (2007)
12. Miller, W., et al.: 28-way vertebrate alignment and conservation track in the UCSC genome browser. Genome Res. 17, 1797–1808 (2007)
13. Elemento, et al.: Reconstructing the duplication history of tandemly repeated genes. Mol. Biol. Evol. 19(3), 278 (2002)
14. Zhang, L., et al.: Greedy method for inferring tandem duplication history. Bioinformatics 19, 1497–1504 (2003)
15. Sammeth, M., Stoye, J.: Comparing tandem repeats with duplications and excisions of variable degree. TCBB 3, 395–407 (2006)
16. Bertrand, D., Lajoie, M., El-Mabrouk, N., Gascuel, O.: Evolution of tandemly repeated sequences through duplication and Inversion. In: Bourque, G., El-Mabrouk, N. (eds.) RECOMB-CG 2006. LNCS (LNBI), vol. 4205, pp. 129–140. Springer, Heidelberg (2006)
17. Ma, J., et al.: The infinite sites model of genome evolution. PNAS 105(38), 14254–14261 (2008)
18. Jiang, Z., et al.: Ancestral reconstruction of segmental duplications reveals punctuated cores of human genome evolution. Nat. Genet. 39(11), 1361–1368 (2007)
19. Zhang, Y., et al.: Reconstructing the evolutionary history of complex human gene clusters. In: Vingron, M., Wong, L. (eds.) RECOMB 2008. LNCS (LNBI), vol. 4955, pp. 29–49. Springer, Heidelberg (2008)
20. Zhang, Y., et al.: Simultaneous history reconstruction for complex gene clusters in multiple species. In: Pacific Symposium on Biocomputing 2009, pp. 162–173 (2009)
21. Schwartz, S., et al.: Human-mouse alignments with BLASTZ. Genome Res. 13(1), 103–107 (2003)
22. Nadeau, J.H., Taylor, B.A.: Lengths of chromosomal segments conserved since divergence of man and mouse. Proc. Natl. Acad. Sci. USA 81(3), 814–818 (1984)
23. Nakayama, J., et al.: Linkage of human narcolepsy with HLA association to chromosome 4p13-q21. Genomics 65, 84–86 (2000)
24. Sollid, L., et al.: Evidence for a primary association of celiac disease to a particular HLA-DQ α/β heterodimer. J. Exp. Med. 169, 345–350 (2000)
25. Haque, A., et al.: HLA class II protein expression in prostate cancer cells. Journal of Immunology 178, 48.22 (2007)
26. Chaudhuri, S., et al.: Genetic susceptibility to breast cancer: HLA DQB*03032 and HLA DRB1*11 may represent protective alleles. PNAS 97, 11451–11454 (2000)

27. Penning, T., et al.: Aldo-keto reductase (AKR) 1C3: Role in prostate disease and the development of specific inhibitors. Mol. Cell. Endocrinol. 248, 182–191 (2006)
28. Crofts, F., et al.: Functional significance of different human CYP1A1 genotypes. Carcinogenesis 15, 2961–2963 (1994)
29. Sato, M., et al.: Genetic polymorphism of drug-metabolizing enzymes and susceptibility to oral cancer. Carcinogenesis 20, 1927–1931 (1999)

Appendix

A Types of Duplication

Let

$$S = s_1 s_2 \ldots s_n$$

be a genomic sequence of length n, where $s_i \in \{A, C, G, T\}$. For any a, b and c such that $1 \leq a \leq b \leq n$ and $1 \leq c \leq n$, a forward duplication that copies the segment $s_a s_{a+1} \ldots s_b$ and inserts it between s_{c-1} and s_c is written $[a, b] + c$. It transforms S into the following sequence

$$S' = s_1 s_2 \ldots s_{c-1}\ \underline{s_a s_{a+1} \ldots s_b}\ s_c s_{c+1} \ldots s_n.$$

If $c = b + 1$, the forward duplication $[a, b] + c$ forms a tandem duplication in the resulting sequence

$$s_1 s_2 \ldots s_{a-1}\ \underline{s_a s_{a+1} \ldots s_b}\ \underline{s_a s_{a+1} \ldots s_b}\ s_{b+1} \ldots s_n.$$

Tandem duplications are observed in many important gene clusters in eukaryotic genomes. If $a < c < b$, then $[a, b] + c$ copies within itself and produces a segment $AABB$ where A and B are non-empty segments in the resulting sequence

$$s_1 s_2 \ldots s_{a-1}\ \underline{s_a \ldots s_{c-1}}\ \underline{s_a \ldots s_{c-1}}\ \underline{s_c \ldots s_b}\ \underline{s_c \ldots s_b}\ s_{b+1} \ldots s_n.$$

A backward duplication inserting the reverse-complement sequence $-s_b-s_{b-1}\ldots-s_a$ between s_{c-1} and s_c is written $[a, b] - c$. If $c = b + 1$, the backward duplication $[a, b] - c$ produces a palindrome. Let $\overline{A}$ denote the reverse complement of sequence A. If $a < c < b$, $[a, b] - c$ produces a segment $A\overline{BA}B$.

B Overlap Relationship

To explain the algorithm, we consider the overlap relationship between two duplication events. Let A be a region of a duplication event and let B be a region of another duplication event in S. A base of location i in S is denoted s_i.

1. If there exist i, j, k, l $(1 \leq i < k \leq j < l \leq n)$ such that

   $$A = s_i s_{i+1} \ldots s_j,\ B = s_k s_{k+1} s_{k+2} \ldots s_l$$

   or vice versa, we say that A *overlaps* B.

2. If there exist i, j, k, l $(i < k < j < l)$ such that

$$\begin{aligned} B &= B_1 B_2 = \underline{s_i s_{i+1} \dots s_k} \ \underline{s_{j+1} s_{j+2} \dots s_l}\ , \\ A &= s_{k+1} s_{k+2} \dots s_j , \end{aligned}$$

   A is said to be *inserted* into B.
3. If there exist i, j, k, l $(i \leq k < j \leq l)$ such that

$$\begin{aligned} A &= s_i s_{i+1} \dots s_k \ \underline{s_{k+1} s_{k+2} \dots s_j} \ s_{j+1} s_{j+2} \dots s_l \\ &= s_i s_{i+1} \dots s_k B_{j+1} s_{j+2} \dots s_l , \end{aligned}$$

   we say that A *contains* B.
4. A region is said to be *disjoint* from other regions if it does not overlap with any other regions, is not inserted into and does not contain any other regions.

C Proof of Theorem 2

We prove it by induction on the number n duplications. The results are trivial when $n = 1$ and 2. Assume it is true for $n \leq k - 1$ and S evolves from a duplication-free sequence T by k duplications $O_1, O_2, ..., O_k$. Let D be the region first selected by the algorithm such that D forms match $M = (R, D)$ or $M = (D, R)$. Then, D does not overlap with other matches and is not contained in any other matches since the out-degree of D is 0 in the constraint graph. Let M be generated by duplication O_i. We consider the following cases.

Case 1. If $i = k$, it means that M is generated by the latest duplication. Assume the resulting sequence is S'' when D is removed from S. If D is a target region of O_k, T transforms into S'' by $O_1, O_2, ..., O_{k-1}$. By induction, the algorithm will reduce S'' to a duplication-free sequence T'' such that T'' transforms into S'' by $k - 1$ duplications. Therefore, T'' transforms into S by k duplications.

If D is a source region of O_k, R is a target region. Since D does not overlap with any other matches and D is not contained in other matches, D cannot be involved in any other duplications. Thus, T includes D, i.e. D is in the duplication-free sequence. Let S' be the resulting sequence after removing D from S and let T' be the resulting sequence after removing D from T and inserting R in the corresponding position of S. By assumption, T' transforms into S' by $O_1, O_2, ..., O_{k-1}$. By induction on S', the algorithm outputs a duplication-free sequence T'' such that it transforms into S' by $k - 1$ duplications. Since S' transforms into S by duplication that creates D, the removal of D guarantees to find a solution of the same number of true events.

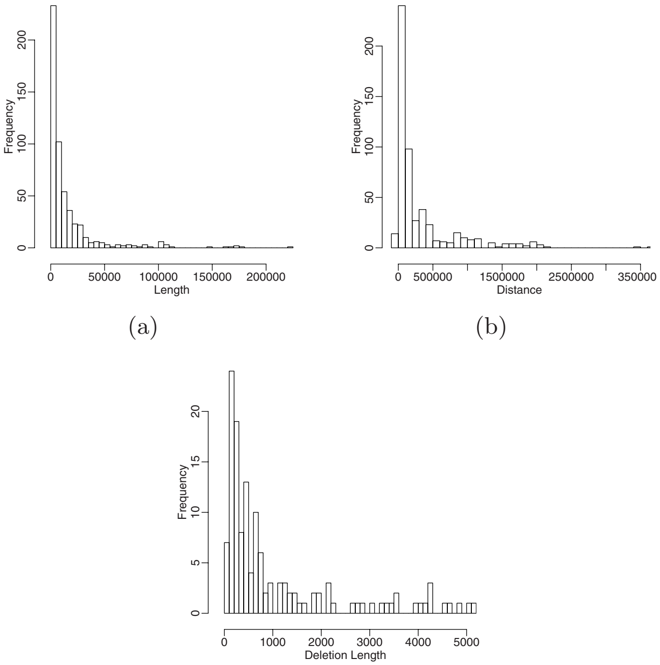
Case 2. If $i < k$, we consider the following sub-cases.

Sub-case 2.1. Suppose both of D and R in M have out-degree 0. If D is removed by the algorithm, all the regions involved in duplications $o_j, i < j \leq k$ do not overlap D. Thus, $o_1, o_2, ..., o_{i-1}, o_{i+1}, ..., o_k, o_i$ also transform T into S. This reduces to Case 1.

Sub-case 2.2. If D is a target region of O_i and its out-degree is 0 and R does not have out-degree 0, D is removed by the algorithm. Then, since D is a target region, the resulting sequence S' after the removal of D in S can be generated from T by $k-1$ duplications $O_1, O_2, ..., O_{i-1}, O_{i+1}, ..., O_k$. Thus, by induction, algorithm reduces S' to a duplication-free sequence T''. Obviously, T'' evolves into S in k duplications.

Sub-case 2.3. If D is a source region of O_i and its out-degree is 0 and R does not have out-degree 0, D is removed by the algorithm. Since D is not contained in other match regions, D is a subsequence of the original duplication-free sequence T. In this case, since the breakpoint uniqueness assumption holds, D is not inserted in any match region, and hence it must be in T. Let T' be the resulting sequence after removal of D and insertion of R in T. Then, T' evolves into S by $k-1$ duplications. By induction, the algorithm identifies a duplication-free sequence T'' that evolves into S by $k-1$ operations. By modifying T'' by inserting R and removing D, we derive a duplication-free sequence that evolves into S in k duplications.

D Figures



(a)

(b)

(c)

Fig. A-1. Distribution of (a) duplication length, (b) distance between the source and target regions for duplications, and (c) deletion length