BIOINFORMATICS ORIGINAL PAPER

Vol. 22 no. 9 2006, pages 1064–1071
doi:10.1093/bioinformatics/btl057

Phylogenetics

Recco: recombination analysis using cost optimization

Jochen Maydt* and Thomas Lengauer

Max-Planck Institut für Informatik, Saarbrücken, Germany

Received on December 8, 2005; revised and accepted on February 13, 2006
Advance Access publication February 17, 2006
Associate Editor: Martin Bishop

ABSTRACT

Motivation: Recombination plays an important role in the evolution of many pathogens, such as HIV or malaria. Despite substantial prior work, there is still a pressing need for efficient and effective methods of detecting recombination and analyzing recombinant sequences.

Results: We introduce Recco, a novel fast method that, given a multiple sequence alignment, scores the cost of obtaining one of the sequences from the others by mutation and recombination. The algorithm comes with an illustrative visualization tool for locating recombination breakpoints. We analyze the sequence alignment with respect to all choices of the parameter α weighting recombination cost against mutation cost. The analysis of the resulting cost curve yields additional information as to which sequence might be recombinant. On random genealogies Recco is comparable in its power of detecting recombination with the algorithm Geneconv (Sawyer, 1989). For specific relevant recombination scenarios Recco significantly outperforms Geneconv.

Availability: Recco is available at http://bioinf.mpi-inf.mpg.de/recco/

Contact: jmaydt@mpi-inf.mpg.de

1 INTRODUCTION

In comparison with tree-based phylogenetic analysis procedures, procedures for analyzing recombination are immature. Recent power studies on recombination detection methods uncovered that it can be hard even to decide whether there is recombination in a set of aligned sequences (Posada and Crandall, 2001; Wiuf *et al*., 2001). The questions, which sequence is the recombinant and where there are recombination breakpoints, are even more challenging.

Methods for analyzing recombination in molecular sequences fall into at least four categories: (1) recombination detection methods, (2) methods for deriving bounds on the number of recombination events (Song and Hein, 2005; Song *et al*., 2005), (3) network methods such as SplitsTree (Huson, 1998) and (4) inference methods based on the coalescent (Kuhner *et al*., 2000; Fearnhead and Donnelly, 2001). Each category is appropriate for different problem settings and can provide independent information on the recombination signal contained in a dataset. As recombination detection programs are conceptually closest to our approach, we limit our comments to this category in the following paragraphs.

In the past 20 years, more than 20 methods have been developed for detecting the presence of recombination in a sequence dataset. More details and an evaluation of their accuracy can be found in recent power studies (Brown *et al*., 2001; Posada and Crandall, 2001; Wiuf *et al*., 2001) and in book chapters (Salminen, 2003; Husmeier and Wright, 2004) dealing with recombination detection. Links to implementations are on the website of D. Robertson (http://bioinf.man.ac.uk/~robertson/recombination/).

The earliest methods use statistical tests for checking whether substitutions in the alignment are non-uniformly distributed, i.e. whether substitutions are significantly clustered owing to recombination, back mutation or other effects. Popular methods using this principle are the maximum χ^2-test (Smith, 1992; Spencer, 2003) and Geneconv (Sawyer, 1989). Geneconv searches for the longest conserved fragment between two sequences and determines whether it is significant. Extensions also allow for including mutations in the fragments. Despite the lack of any explicit model of evolution, substitution distribution methods are quite competitive (Posada and Crandall, 2001), with Geneconv performing as one of the best methods.

Many other methods detect a change of the phylogenetic distance signal in adjacent areas of the alignment. Some popular methods are PLATO (Grassly and Holmes, 1997), TOPAL (McGuire and Wright, 2000), PhyPro (Weiller, 1998) and SimPlot (Lole *et al*., 1999). These methods either use a global reference tree or a sliding window to detect local changes in the topology of the phylogenetic distance signal. A global reference tree is problematic for strong recombination signals, as a dominant phylogenetic tree cannot be identified anymore and artifacts are introduced into the tree (Schierup and Hein, 2000). A fixed window size determines the trade-off between localizing the breakpoints accurately and the ability to correctly infer recombination. Even though all these approaches use a model of (tree-like) evolution, they lack a model for recombination. Consequently, they only look for indirect evidence of recombination and thus may falsely detect recombination.

Another group of methods is more closely related to the coalescent framework as they infer a restricted version of the evolutionary history subject to recombination. RecPars (Hein, 1993) allows for a different tree-like evolutionary history at each position and tries to heuristically minimize the cost for substitution at each position under the associated tree as well as the cost for topology changes along the sequence of trees. Thus, RecPars does away with the window-size parameter, but adds recombination and mutation cost parameters. Husmeier and McGuire (2003) translate the idea of RecPars into a statistical framework and maximize the likelihood of topology changes and mutations. While being accurate, a major drawback of this approach is the high computational effort that makes this method inapplicable for datasets with more than a few sequences.

We present Recco, a fast and simple method for detecting recombination in a set of sequences and locating putative recombination breakpoints that is based on cost minimization and dynamic programming. The basic idea is to construct each sequence in the

*To whom correspondence should be addressed.

alignment (temporarily considered the putative recombinant) in turn, from the other sequences in the alignment using only the mutation and recombination operators. The output of Recco bears some resemblance with SimPlot (Lole *et al*., 1999) and represents local sequence similarity of the putative recombinant with the other sequences. In contrast to SimPlot, there is no need for a sliding window and hence no limitation of spatial resolution. The minimum cost solution identifies the best recombination breakpoints and also the parental sequences. Recco has only two tunable parameters, recombination and mutation cost. In practice, we only need to consider a single parameter α representing the cost of mutation relative to recombination. We present an approach for finding the values for α at which the solution changes structurally. This can be condensed further into a single indicator for the presence of recombination in the alignment.

The dynamic program we use is based on the work of Kececioglu and Gusfield (1998). It employs insertion, deletion, recombination and mutation for producing a single sequence from two other sequences. Our method is a restricted version of the dynamic programs introduced in Lajoie and El-Mabrouk (2005) and Spang *et al*. (2002). Lajoie and El-Mabrouk use recombination, mutation and gene conversion in order to explain a single sequence from an alignment of other sequences. Spang *et al*. (2002) allow for insertion, deletion, recombination and mutation in order to derive a single sequence from an alignment of other sequences. Even though such a more general formulation is usually preferable, our visualization technique introduced in Section 2.3.1 depends on the restrictions that we have imposed. More general formulations would need at least three dimensions for visualization. The parametric analysis which we describe in Section 2.3.2 is also unique to our work, but in principle applicable to other methods as well. For example, RecPars could benefit from this type of analysis.

2 METHODS

In order to illustrate our method, we use the same sequence data throughout the discussion. The data were generated by the program Seq-Gen (Rambaut and Grassly, 1997) using the genealogy depicted in Figure 1. First we eliminate the sequence R2 from the dataset. Then the dataset does not contain two similar recombinants which simplifies the problem for purposes of exposition. The general problem is discussed in the results section. Hence, sequence R1 is the only recombinant and is derived from an ancestor of A1 to the left and an ancestor of B1 to the right. The breakpoint is located in the middle of the alignment. The total length of the alignment is 100 nt. We used the Jukes–Cantor model of nucleotide evolution (Jukes and Cantor 1969) with an average number of 0.25 substitutions per site from the root to any of the tips of the genealogy. The full dataset including sequence R2 is analyzed in the Results section.

2.1 Recombination analysis through cost minimization

Let A be a multiple sequence alignment with M rows and N columns, with A_i representing the i-th sequence and A_{ip} denoting the p-th nucleotide of the i-th sequence. Let $S = s_1, \ldots, s_N$ be the sequence of the putative recombinant and define $m(i, p)$ as the cost of substituting A_{ip} with s_p. Recombination cost is measured by $r(i, j, p)$ and represents the cost of recombining A_i on the left side of position p with A_j on the right side, hence resulting in the sequence $A_{i1}, \ldots, A_{i,p-1}, A_{jp}, \ldots, A_{jN}$. Usually it makes sense to set $r(i, i, p) = 0$ for all i and p, as this does not represent a visible recombination. Our goal is to find a sequence of mutation and recombination events that produce S from the sequences in A. In the formal model we use a decision variable $k_p \in \{1, \ldots, M\}$ to determine the sequence of A explaining s_p, so that S is explained by $A_{k_1,1}, \ldots, A_{k_N,N}$. The most parsimonious solution then minimizes the following cost function with respect to $\mathbf{k} = (k_1, \ldots, k_N)$:

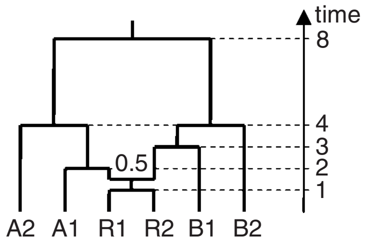

Fig. 1. The genealogy used for generating the example dataset for Figures 3, 4 and 6. R1 and R2 are created by recombining sequence A1 to the left with sequence B1 to the right. The breakpoint occurs at the middle of the sequence. Time is scaled so that there are 0.25 mutations per position from the root to a tip of the tree, on average.

$$\min_{\mathbf{k}} \varphi_\alpha(\mathbf{k}), \tag{1}$$

$$\text{where } \varphi_\alpha(\mathbf{k}) = (1-\alpha)\sum_{p=2}^{N} r(k_{p-1}, k_p, p) + \alpha \sum_{p=1}^{N} m(k_p, p). \tag{2}$$

A solution $\mathbf{k}$ infers a mutation event for each position p where $A_{k_p,p} \neq s_p$ and a recombination event for each p where $k_{p-1} \neq k_p$. The parameter $\alpha \in [0,1]$ weighs mutation against recombination cost and accounts for the fact that recombination can always be explained by mutation. If α is close to zero the solution favors mutations. For $\alpha = 1$ recombination costs nothing and the optimal solution uses the nucleotide A_{ip} most similar to s_p at each position p. In the following, we use unit cost for recombination and mutation, i.e. $r(i, j, p) = \delta(i, j)$ and $m(i, p) = \delta(A_{ip}, s_p)$ where δ is the Kronecker delta function.

While α selects the global preference for recombination against mutation, $m(i, p)$ and $r(i, j, p)$ can be adapted to model local changes in recombination and mutation frequency and lead to a simple strategy for incorporating prior knowledge on the recombination and mutation process. A mutation hotspot or variable region p' might have a lower cost $m(i, p')$ for substitution than other positions p. Also, mutation to a rare nucleotide can incur a higher cost than mutation to a common nucleotide, thereby accommodating nucleotide composition bias. The recombination cost parameter $r(i, j, p)$ can also vary among positions and capture the lower cost at recombination hotspots. However, it is also possible to model much more complex effects, as $r(i, j, p)$ also depends on the sequence pair (i, j). For example, several mechanisms proposed for HIV recombination (Negroni and Buc, 2001) depend on the donor sequence i, on the acceptor sequence j or on the pair of parental sequences (i, j) that produce the recombinant. Such effects can easily be accounted for by increasing or decreasing the respective recombination cost $r(i, j, p)$.

2.2 Formulation as a dynamic program

Despite its expressive power, the proposed optimization problem can be solved efficiently. As the cost function (1) is separable in k_p, we can minimize it by dynamic programming, see Figure 2 for the corresponding tableau. To be more specific, define a subproblem of (1) by only considering the columns $1, \ldots, p$ of the alignment, with the additional constraint that solutions end using some sequence k_p. The minimal cost solution of this subproblem is the minimal cost for generating S via mutation and recombination up to and including position p, using all sequences in A and ending in sequence k_p:

$$f_{k_p,p} = \min_{k_1,\ldots,k_{p-1}} \left\{ (1-\alpha)\sum_{q=2}^{p} r(k_{q-1}, k_q, q) + \alpha \sum_{q=1}^{p} m(k_q, q) \right\} \tag{3}$$

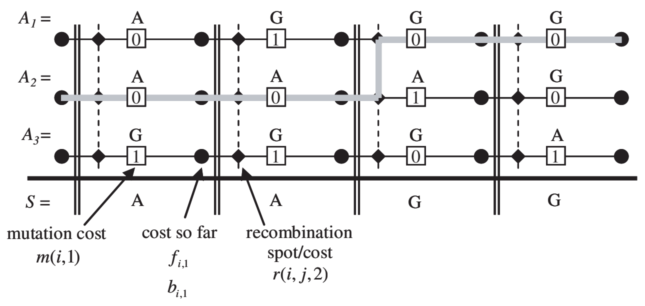

Fig. 2. The structure of the dynamic program tableau. The circles represent the values f_{ip} and b_{ip} of the tableau. The diamonds and the boxes specify the place at which recombination and mutation occur, respectively. Inside the boxes, the mutation cost relative to the putative recombinant S is given. The vertical double lines separate the stages of the dynamic program and also visualize the correspondence of the actual sequence A_{ip} to the values f_{ip} and b_{ip}. An example solution $\mathbf{k} = (2,2,1,1)$ explaining the putative recombinant S by A_1 and A_2 is marked with a thick grey line and involves one recombination and no mutation event.

This can be rewritten as the following forward recurrence for all $p = 1, \ldots, N$ as the optimal value of k_p does not depend on $k_1, \ldots, k_{p-2}$:

$$\begin{aligned} f_{k_p,p} &= \min_{k_{p-1}} \left\{ f_{k_{p-1},p-1} + (1-\alpha) r(k_{p-1}, k_p, p) + \alpha m(k_p, p) \right\} \\ f_{j,0} &= 0 \quad \text{for all } j. \end{aligned} \tag{4}$$

Each f_{jp} represents the minimal cost for the subproblem restricted to columns $1, \ldots, p$. The minimal cost for the complete problem is then $\min_j f_{jN}$ and the optimal solution can be retrieved by backtracing. The time complexity is $O(NM^2)$ for the forward run as there are N stages with each stage performing M^2 operations. The backtracing step needs $O(NM)$ operations and $O(NM)$ space as the tableau f_{jp} has to be stored and each value is accessed exactly once.

Alternatively to the forward recurrence, we can also use the backward recurrence for $p = N-1, \ldots, 0$ and describe the minimal cost of a solution generating S backwards, starting from position N up to and excluding position p ending in sequence k_p. The backward recurrence is

$$\begin{aligned} b_{k_p,p} &= \min_{k_{p+1}} \left\{ b_{k_{p+1},p+1} + (1-\alpha) r(k_p, k_{p+1}, p+1) + \alpha m(k_{p+1}, p+1) \right\} \\ b_{i,n} &= 0 \quad \text{for all } i. \end{aligned} \tag{5}$$

2.3 Sensitivity analysis

Usually, we are not only interested in the optimal solution itself, but also in the sensitivity of the solution with respect to small changes of the involved variables. Sensitivity is commonly defined in one of two ways: sensitivity with respect to small changes of the optimal solution or sensitivity with respect to small changes of input parameters, such as α, $m(i, p)$ and $r(i, j, p)$ in our case. A sensitivity analysis with respect to the optimal solution, subsequently called 'analysis of the solution', studies the cost surface in the neighborhood of the optimal solution. The second approach, subsequently called 'parametric analysis', quantifies how the optimal solution depends on a set of parameters. More exactly, it partitions the space of parameter settings into regions that yield the same optimal solution and assesses the impact of uncertainty of parameters, e.g. due to measurement or estimation errors, on the structure of the optimal solution. Both types of analyses have been applied separately to the problem of sequence alignment; see e.g. Mevissen and Vingron (1996) and Zimmer and Lengauer (1997). In the following we use both types of analysis.

2.3.1 Analysis of the solution. In the following we transfer an idea by Mevissen and Vingron (1996) from traditional sequence alignment to our problem.

For each position p and each sequence i we compute the minimum cost c_{ip} of a solution that runs through nucleotide A_{ip}. In other words, we constrain $\mathbf{k}$ to solutions with $k_p = i$. We can compute the minimal cost of the constrained problem by adding the values of the forward and backward tableaus $c_{ip} = f_{ip} + b_{ip}$, since f_{ip} is the minimum cost of generating the recombinant up to and including position p ending in sequence i, and b_{ip} is the minimum cost starting from position N up to and excluding position p ending in sequence i (Fig. 2).

We define a robustness measure r_{ip} by $r_{ip} = \min_{j \neq i} \{c_{jp}\} - c_{ip}$. In words, r_{ip} is the sacrifice in cost that is incurred by forcing a solution that avoids position p in sequence i in our alignment. If r_{ip} is small then the choice of position p in sequence i does not buy us much in terms of alignment cost. Larger values of r_{ip} indicate more robust sequence positions. The values c_{ip} and r_{ip} can be computed in time $O(MN)$ for all combinations of p and i if the forward and backward tableau f_{ip} and b_{ip} are given.

Both values, c_{ip} and r_{ip}, are useful in our scenario. c_{ip} is particularly attractive for visualizing the optimal solution of the dynamic program. All nucleotides A_{ip} that are part of an optimal solution have the same, minimal c_{ip}. We can visualize these by coloring the background of the alignment A_{ip} according to the values c_{ip} (Fig. 3a). For positions at which the solution is uncertain or close to a recombination breakpoint, low values of c_{ip} occur for several sequences to indicate the ambiguity. On the other hand, if we ask how robust the choice of a specific nucleotide A_{ip} is, we can rely on the values of r_{ip}. Regions that are uncertain are characterized by a lower robustness of the optimal solution (Fig. 3b).

2.3.2 Parametric analysis. Even though the analysis of the previous section analyzes the cost surface of solutions close to the minimum solution and also results in an appealing visual representation, it cannot capture the uncertainty caused by varying the input parameters. In particular, for many pathogens we do not have reliable estimates of the recombination rate and cannot set the parameter α appropriately. But even if we know the recombination rate and the appropriate setting of α, it is still informative to find the region of α for which the optimal solution does not change structurally. In the following we study how $\phi_\alpha(\mathbf{k})$ changes as α is varied, both for a fixed solution $\mathbf{k}$ and for $\mathbf{k}$ subject to optimization.

If $\mathbf{k}$ is fixed, $\phi_\alpha(\mathbf{k})$ is linear in α and has the derivative

$$\frac{\partial \varphi_\alpha(\mathbf{k})}{\partial \alpha} = -R + M,$$

where

$$R = \sum_{p=2}^{N} r(k_{p-1}, k_p, p) \text{ and } M = \sum_{p=1}^{N} m(k_p, p). \tag{6}$$

R and M are the total cost of recombination and mutation not weighted by $(1-\alpha)$ or α. If $\mathbf{k}$ is also subject to optimization, the minimal cost is a piecewise-linear concave function $g(\alpha) = \min_{\mathbf{k}} \varphi_\alpha(\mathbf{k})$. Each linear piece corresponds to a fixed set of minimum cost solutions. This set can be retrieved by the dynamic program and only changes at the edges of the piecewise-linear cost function. Hence, it is simple to retrieve all minimum cost solutions for $\alpha \in [0,1]$, given the parametric cost curve g.

In order to compute the parametric cost curve g, we use the algorithm of Eisner and Severance (1976). In order to apply this algorithm, we need a method for computing the optimal cost and the derivative at a given point α. In our case it is simple to compute the minimal cost with the dynamic program. To get the derivative, we extended the backtrace algorithm to compute R and M.

The result of the parametric optimization for our example dataset is shown in Figure 4, where each sequence of the alignment was taken as the putative recombinant, in turn. Each linear piece of a cost curve is defined by some

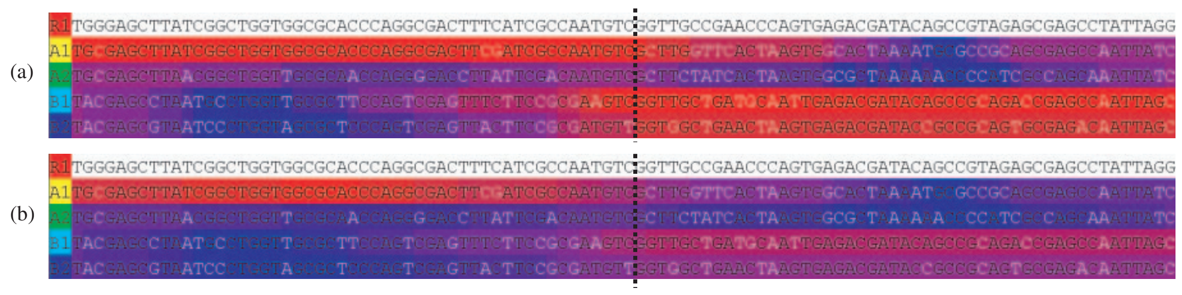
(a)
(b)
R1
A1
A2
B1
B2
TGGGAGCTTATCGGCTGGTGGCGCACCCAGGCGACTTTCATCGCCAATGTCGGTTGCCGAACCCAGTGAGACGATACAGCCGTAGAGCGAGCCTATTAGG

Fig. 3. Result of the forward–backward algorithm for explaining sequence R1. We used unit cost for recombination and mutation and set parameter α to 0.2. Thus, each recombination involves the same cost as four mutations. The background color represents the value of (**a**) the cost measure c_{ip} and (**b**) the robustness measure r_{ip}. Red represents low cost and high robustness, while blue corresponds to the opposite. A white foreground color highlights mutations with respect to R1. The optimal solution in (a) is a path from left to right using only cost-optimal nucleotides, and switches from sequence A1 to sequence B1 at about the middle of the alignment. The true breakpoint in the middle of the alignment is marked by a dotted vertical line.

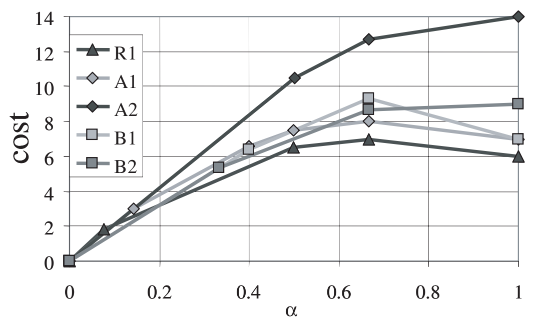

Fig. 4. The parametric cost curves for each sequence in the alignment of Figure 3. As α changes from 0 to 1, the cost for mutation weighted by α increases and the cost for recombination weighted by $(1-\alpha)$ decreases. The cost curve of R1 shows the highest benefit from recombination. The first recombination is introduced into the solution at $\alpha = 0.077$. This solution remains optimal up to $\alpha = 0.5$. Other sequences, such as B1, B2 or A2 do not show a high preference for recombination.

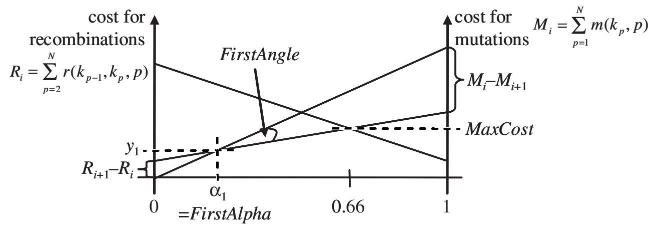

Fig. 5. Schema of a parametric cost curve together with the extracted features. The feature MaxSavings is not shown and is computed as $(M_1 - M_2)/(R_2 - R_1)$.

values of R and M and has the functional form $(1-\alpha)R + \alpha M$. Hence, we can read-off the values of R and M as the abscissa for $\alpha = 0$ and $\alpha = 1$, respectively. For example, the first linear piece of each cost curve intersects the origin and corresponds to a solution with $R = 0$, that means without recombination. All other solutions contain at least one recombination. The unweighted cost for recombinations R is increasing with increasing α, as mutation cost grows relative to recombination cost. The transition from a solution containing only mutations to a solution with at least one recombination is of particular interest. If the transition happens at a small value of α, recombinations are introduced, even though their weighted cost $(1-\alpha)r(i, j, p)$ is high relative to the cost of mutations. To be more specific, if recombinations and mutations have unit cost, each recombination reduces the number of mutations in the solution by $(1-\alpha)/\alpha$. For instance, for sequence R1 in Figure 4, we have $\alpha = 0.077$ and the sole recombination in the solution associated with $\alpha \in [0.077, 0.5]$ removes about $0.923/0.077 \approx 12$ mutations. Hence, a small α value is indicative for a true recombinant and can be used for filtering interesting sequences from the dataset. In the following section we use a similar property in an automated method for detecting recombinant sequences.

2.4 Recombination detection

The previous section introduced sensitivity analysis as a means of investigating a sequence dataset manually and pointing out single recombination events. In the following, we focus on extending this type of analysis quantitatively, such that it can also be used in an automated procedure discerning recombinant from non-recombinant datasets. This does not only yield P-values for the presence of recombination, but also forms the basis for the power study in Section 3. A quick overview of the features we introduce in the following sections is given in Figure 5.

Given the segments of a cost curve ordered by increasing α, let R_i and M_i, $i = 1, \ldots, K$ be the unweighted cost of recombination and mutation for the i-th linear segment. The first feature that we introduce is the amount of mutation cost saved per unit of recombination cost. More exactly, $R_{i+1} - R_i$ is the additional amount of recombination cost incurred by solution $i+1$, and $M_i - M_{i+1}$ is the amount of mutation cost saved. The quotient $\text{Savings}_i = (M_i - M_{i+1})/(R_{i+1} - R_i)$ quantifies the additional amount of mutation cost saved by each additional unit of recombination cost, where Savings_i is decreasing with increasing i. Consequently, we define our first feature to be

$$\text{MaxSavings} = \max_i \left(\frac{M_i - M_{i+1}}{R_{i+1} - R_i} \right) = \frac{M_1 - M_2}{R_2 - R_1} = \text{Savings}_1 \qquad (7)$$

Other features do not perform as well for recombination detection or are equivalent to MaxSavings. One example is FirstAlpha, the smallest α-value for which a recombination is introduced. Using simple geometric arguments it can be shown that $(1 - \text{FirstAlpha})/\text{FirstAlpha} = \text{Savings}_1$. This is a strictly monotonic transformation as $\text{FirstAlpha} \in [0,1]$, and thus a P-value based on FirstAlpha is exactly the same as one based on MaxSavings. Therefore, we do not consider the feature FirstAlpha in our analysis. Another feature that is related but not equivalent to MaxSavings is FirstAngle, the angle in radians between the first and second linear segment of a cost curve (Fig. 5). A large angle points to a significant decrease of mutation cost and an increase of recombination cost. The last feature we propose is MaxCost, the maximum that the cost curve attains. This feature measures how hard it is to derive the putative recombinant from the other sequences in the alignment. The value of MaxCost depends mostly on the time when the sequence diverged from the other sequences in the dataset. A sequence that diverged early in history tends to have a higher value of MaxCost than a sequence that diverged only recently.

In general we cannot rely on absolute feature values in identifying recombination. The features depend on parameters of the dataset, such as sequence diversity or sequence length. Hence, we use a permutation test to obtain *P*-values instead of absolute feature values for each sequence. The *P*-value is computed by comparing the observed feature value with the feature value expected under the null-hypothesis that no recombination occurred. More exactly, we permute the columns of an alignment 100 times in order to generate independent samples of datasets with the same sequence diversity, but without a recombination signal. In the following, a sequence with a *P*-value ≤ 0.05 is assumed to be recombinant. As the number of sequences is usually small, we do not consider any correction for multiple testing.

The *P*-values suggest whether a certain sequence is recombinant or not. They do not imply that the whole dataset contains a significant amount of recombination. In order to test the latter hypothesis we have to condense the *P*-values or feature values for the whole dataset into a single *P*-value, while controlling the level of false positives. A simple choice is to use the smallest *P*-value over all sequences in the alignment and to apply a Bonferroni correction for multiple testing. This is very conservative and frequently reduces power. Alternatively, we can compute the minimum, or the most extreme, feature value over all sequences of a dataset and repeat this step for every permutation. The resulting *P*-value for the whole dataset still controls the false positive rate nicely and performs better than the Bonferroni correction. Again, a significance cutoff of 0.05 is used to indentify recombination.

2.5 Detection of the breakpoint

The backtrace contains information on the recombination breakpoints of a solution **k**. However, the inferred breakpoints depend on the selected sequence S and on α. In the following, we present a method that assigns *P*-values to each position without depending on S or α.

We address the dependence on α first. The method is based on a recombination profile p_j, $j = 1, \ldots, N$, that stores how much mutation cost can be saved per unit of recombination cost if we introduce a recombination at position j in the solution explaining a specific sequence S. More exactly, for each linear piece $i = 1, \ldots, K$ of the parametric cost curve we denote the set of breakpoints that occur in any optimal solution of i as B_i. Then $p_j = \max_{i=2,\ldots,K} \{0\} \cup \{\text{Savings}_{i-1} \mid j \in B_i\}$, where Savings_{i-1} is the savings value for solution i. p_j thus stores the maximum Savings value that any breakpoint at position j can achieve. To obtain *P*-values for each breakpoint position, we use a similar approach as in Section 2.4: by computing recombination profiles for column permutations of the original alignment, we can estimate the distribution of p_js for each position j. The result is a *P*-value for recombination at each position and for each sequence in the alignment.

To detect breakpoints without referring to a specific sequence S, we compute the maximum of p_j over all sequences in the alignment for each position j (Fig. 6). Computing *P*-values for the aggregated recombination profile is then analogous to the procedure for a single sequence. Figure 6 shows that a fixed Savings value is more significant for a position close to the boundary than for a position in the middle of the alignment. This is due to the fact that recombinations close to the boundary of the alignment exchange fewer nucleotides and thus cannot achieve high values of Savings.

Detecting the breakpoint position entails a multiple testing problem. Each position has an expected false detection rate of 5% and even our short example alignment could result in five falsely detected breakpoints, on average. Therefore, we would have to adjust for multiple testing, e.g. using the Bonferroni or false discovery rate method (Benjamini and Hochberg, 1995). In order to do so, we would have to perform, say, 10 000 permutations per dataset in order to obtain accurate *P*-values after adjustment. This is computationally forbidding in the context of a power study. Thus, for this validation we only consider the basic methodology without adjustment for multiple testing.

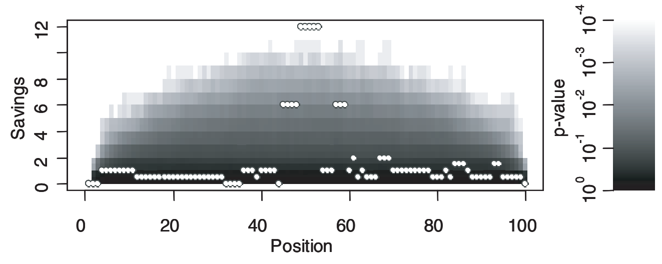

Fig. 6. The recombination profile for our example dataset is shown by white circles. The background visualizes the *P*-value of a Savings value for each position, computed from 10^4 permutations. The (true) breakpoint in the middle of the alignment is highly significant.

3 RESULTS

Validation is difficult for recombination detection programs and even harder for programs finding the recombinant sequences. One reason is that real datasets either show a recombination signal that is easy to detect, or there is still uncertainty about whether recombination actually took place, see e.g. Posada (2002). In comparison, synthetic datasets have many advantages, such as an unlimited amount of datasets and total control over the simulated recombination scenario. Synthetic datasets are usually generated by one of two approaches. The first is to randomly draw an ancestral recombination graph given several population genetic parameters (Griffiths and Marjoram, 1997) and then simulate sequences along the resulting genealogy. Alternatively, we can specify a fixed genealogy for simulating sequences. The first approach works well for comparing methods regarding their power of detecting recombination. The second approach allows studying the power of identifying recombinant sequences or recombination breakpoints as both are known. In the following we use both kinds of synthetic datasets in order to compare our method with existing recombination detection programs.

3.1 Random genealogies

In analogy to the study of Posada and Crandall (2001), we used the coalescent to generate 1000 sequence datasets for different parameter combinations. For all simulations we chose the effective population size $N_e = 1000$, the sequence length of $N = 1000$ nt, a sample size of $M = 10$ sequences and the Jukes–Cantor substitution model. We let the scaled recombination rate ρ and the scaled mutation rate θ vary such that $\rho \in \{0, 1, 4, 16\}$ and $\theta \in \{10, 30, 50, 100, 200\}$. For diploid populations the coalescent defines $\theta = 4N_e\mu N$ and $\rho = 4N_e rN$, where μ is the mutation rate per generation and loci, and r is the recombination rate per generation and loci. Thus, our simulations correspond to $r \in \{0, 2.5 \times 10^{-7}, 1 \times 10^{-6}, 4 \times 10^{-6}\}$ and $\mu \in \{2.5 \times 10^{-6}, 7.5 \times 10^{-6}, 1.25 \times 10^{-5}, 2.5 \times 10^{-5}, 5 \times 10^{-5}\}$. To obtain a measure of the false detection rate, we also simulated data without recombination and different mutation rates and rate variations among sites. Rate variation was modeled using the gamma distribution with shape parameter γ. The gamma distribution was approximated with eight rate categories (Yang, 1996). The simulations without recombination used the same values for θ and $\gamma \in \{0.01, 0.5, 2, \infty\}$. A higher value of γ represents a lower amount

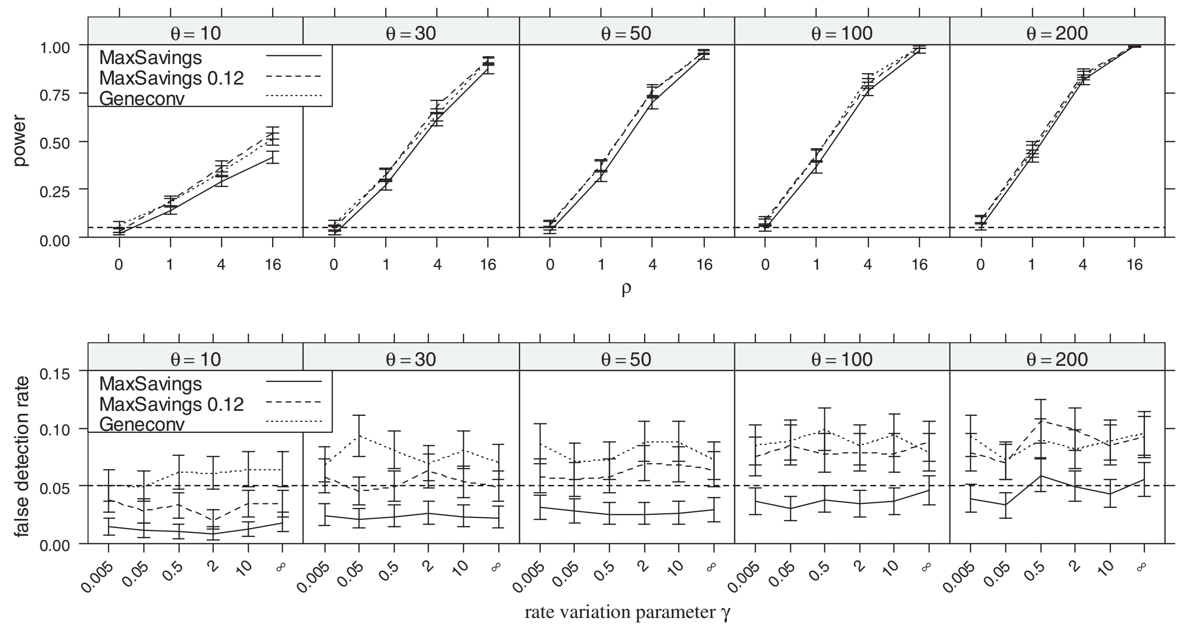

Fig. 7. The probability of detecting recombination for different parameter settings and methods. The top row shows the power of detecting recombination as the mutation rate θ and recombination rate ρ are varied. The bottom row shows the false detection rate for different settings of rate variation γ and mutation rate θ. MaxSavings 0.12 is the same method as MaxSavings, but classifies datasets with a P-value < 0.12 as recombinant. It has a similar or lower false detection rate than Geneconv and a comparable power to detect recombination.

of rate variation, and $\gamma = \infty$ denotes no rate variation at all. All simulations were carried out using treevolve (Grassly *et al.*, 1999). Treevolve did not terminate for recombination rates $\rho \geq 32$, thus we could not cover the whole range of parameters as in Posada and Crandall (2001). Moreover, treevolve generated datasets that are different from the ones used in the original study. For comparison, we report the results on our datasets for Geneconv (Sawyer, 1989), one of the best performing methods in the original study. The parameters of Geneconv were set as in the original study, i.e. using global permutation values and a gscale value of 0. A gscale value of 0 does not allow for mismatches in the conserved fragments between sequences and performed best for detecting recombination in random genealogies.

The probability of detecting recombination for Geneconv and the MaxSavings feature of Recco is depicted in Figure 7. The other features we have proposed did not perform well in this scenario and failed to detect recombination in many cases. Geneconv has a higher probability of detecting recombination than MaxSavings, but does not control the false positive rate of 5% for high mutation rates. Our permutation-based approach controls the false detection rate always to a level below 5%. In order to compare the two methods, we have adjusted the MaxSavings feature such as to classify a dataset as recombinant if the P-values do not exceed 0.12. This setting increases the false detection rate to a level slightly below that of Geneconv, but also increases the power to a comparable level. Thus, our method performs as well as Geneconv in this scenario. However, we have reasons to believe that the power of Recco is reduced by a simple limitation: if two recombinants are similar, they are explained by mutation instead of recombination. This is investigated in more detail in the next subsection.

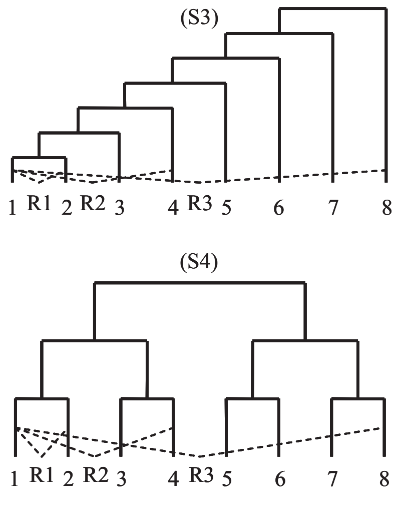

Fig. 8. The genealogies used to simulate data for scenarios (S3) and (S4). Recombinations are marked with dotted lines and always occur in the middle of the sequence. More details on the parameters can be found in Table 1.

3.2 Fixed recombination scenarios

Fixed recombination scenarios can be used for studying the power of identifying the recombinant or the recombination breakpoint, as the recombination scenario is known in detail. We use a total of four recombination scenarios. (S1) and (S2) are generated using the genealogy in Figure 1, but (S1) does not contain sequence R2. (S3) and (S4) were originally introduced in Holland *et al.* (2002) and are given in Figure 8. As all scenarios only contain recent

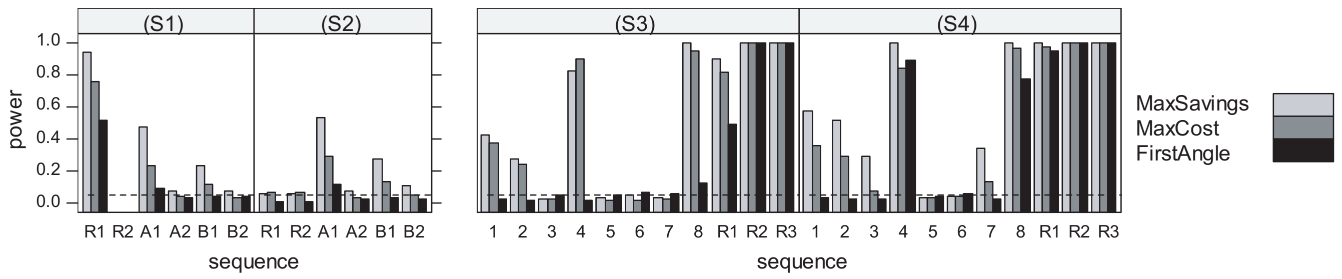

Fig. 9. The probability to detect a sequence as recombinant for different recombination scenarios. For (S1) sequence R2 is missing, and R1 has a very high chance to be detected as recombinant. R2 is included in (S2) and both recombinants R1 and R2 are not detected anymore as recombinant. Only the sequences directly involved in recombination, A1 and B1, have a chance to be detected as recombinant. The scenarios (S3) and (S4) show a similar behavior. Interestingly, FirstAngle has a lower chance to classify parental sequences involved in recombination as recombinants.

Table 1. The power to detect recombination for fixed recombination scenarios

	Sequence length (nt)	Tree height	Mutation model	ts/tv ratio	Geneconv	Max-Savings	First-Angle	Max-Cost
(S1)	100	0.25	JC	—	0.518	0.467	0.012	0.106
(S2)	100	0.25	JC	—	0.472	0.875	0.042	0.572
(S3)	1000	0.15	K2P	2	1.000	1.000	0.303	0.933
(S4)	1000	0.15	K2P	2	1.000	1.000	0.950	0.988

Tree height is given in mutations per site from the root to any tip. The mutation model is either JC (Jukes and Cantor, 1969), or K2P (Kimura, 1980). ts/tv ratio is the transition to transversion ratio.

recombinations, the recombination signal has a very simple structure and should be fairly easy to detect. For each scenario we simulated 1000 alignments using Seq-Gen (Rambaut and Grassly, 1997). We then applied Recco and computed the P-values for recombination per dataset, per sequence and for each position. We observed that in this scenario Geneconv performed much better using a gscale value of 1, i.e. allowing for mismatches in the conserved fragments. Thus, we only report the results using a gscale value of 1 for Geneconv. The result for recombination detection and the parameters used in the simulation are given in Table 1. The most predictive feature for recombination detection in most scenarios was MaxSavings. MaxSavings performed slightly worse than Geneconv only for its worst-case scenario (S2), where the recombinants are very similar. As (S1) and (S2) use a very short sequence length, we repeated this experiment with a sequence length of 1000 nt and one-tenth of the mutation rate. The results did not differ qualitatively and MaxSavings still performed best for scenarios (S2), (S3) and (S4). Recco was also quite fast: a full analysis with 100 column permutations of a dataset from scenario (S3) took ~44 s on an AMD Opteron 2.4 GHz. This included computing the parametric cost curve 101 times for each of the 11 sequences with 1000 nt, constructing breakpoint profiles and summarizing the results as P-values.

Finding the recombinant sequence is more difficult (Fig. 9), particularly because it is very hard to discriminate between the true recombinant and sequences involved in the recombination. This is an intrinsic problem. As expected, the recombinant sequences had the highest probability of being detected as recombinant for (S1),

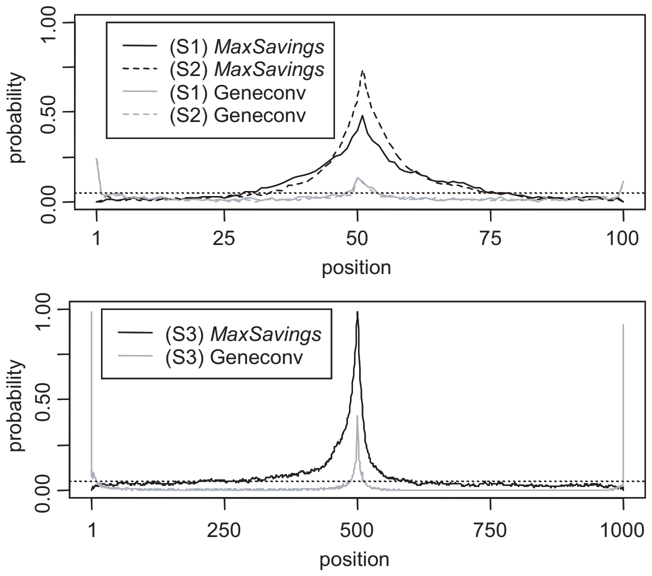

Fig. 10. The probability to detect a recombination breakpoint at each position for different scenarios. Recco has a higher sensitivity, but also reports more spurious findings.

(S3) and (S4). For (S2) the parental sequences A1 and B1 are the only sequences that are detected as recombinant. R1 and R2 are masked as they are very similar to each other. Consequently, the evidence for recombination is derived from sequences A1 and B1, and not from R1 for scenario (S2). Another interesting aspect is that the features MaxCost and FirstAngle performed quite well, despite their failure to detect recombination for random genealogies. Particularly FirstAngle seems to discriminate better between true recombinant and parental sequences. MaxCost is almost as good as MaxSavings here, and achieved a false detection rate <2.5% in all datasets of Subsection 3.1 (data not shown).

To evaluate breakpoint detection performance, we computed the probability of detecting a recombination breakpoint at each position for Geneconv and for the MaxSavings feature of Recco. A breakpoint for Geneconv was defined as the start or end position of a gene conversion fragment with a significant P-value. The result for detecting recombination breakpoints in scenarios (S1), (S2) and (S3) is given in Figure 10. (S4) shows a very similar behavior to (S3). As a side effect of our breakpoint definition for Geneconv there is a high chance of identifying the first or last positions of an alignment as a breakpoint. This should be ignored. Geneconv has a smaller false detection rate than the MaxSavings feature of Recco, but fails to detect the true breakpoint in many cases. The scenarios (S1) and (S2) have a low diversity and are very difficult for Geneconv. Recco is more suitable for these datasets and may assign a highly significant P-value to breakpoints even after correcting for multiple testing (Fig. 6). However, Recco's false detection rate is too high for long sequences if multiple testing is not accounted for.

4 CONCLUSION

We have introduced the new fast method Recco for analyzing sequences subject to recombination. The MaxSavings feature of Recco performs as well as or better than many other methods for recombination detection. While the underlying dynamic program is quite simple, the output of Recco is enhanced considerably by succeeding sensitivity analysis. Sensitivity analysis on the solution provides an intuitive visualization of the solution and alternative recombination hypotheses. We have also introduced a sensitivity analysis on the input parameter α, which controls the ambiguity between mutation and recombination. We believe that this type of analysis can be of benefit in other approaches to recombination detection as well, e.g. RecPars (Hein, 1993). A major limitation of our approach is that two similar recombinant sequences mask each other and are not detected as recombinant. We believe that resolving this issue will lead to an algorithm that is even more powerful. A manual work-around is to iteratively remove the closest sequence to the sequence under investigation until it is detected as recombinant.

ACKNOWLEDGEMENTS

The authors would like to thank Andreas Meyerhans for helpful discussions on the requirements of a recombination analysis tool and on the biology of the recombination process in HIV.

Conflict of Interest: none declared.

REFERENCES

Benjamini,Y. and Hochberg,Y. (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. R. Stat. Soc. Ser. B*, **57**, 289–300.

Brown,C.J. *et al.* (2001) The power to detect recombination using the coalescent. *Mol. Biol. Evol.*, **18**, 1421–1424.

Eisner,M.J. and Severance,D.G. (1976) Mathematical techniques for efficient record segmentation in large shared databases. *J. ACM*, **23**, 619–635.

Fearnhead,P. and Donnelly,P. (2001) Estimating recombination rates from population genetic data. *Genetics*, **159**, 1299–1318.

Grassly,N.C. *et al.* (1999) Population dynamics of HIV-1 inferred from gene sequences. *Genetics*, **151**, 427–438.

Grassly,N.C. and Holmes,E.C. (1997) A likelihood method for the detection of selection and recombination using nucleotide sequences. *Mol. Biol. Evol.*, **14**, 239–247.

Griffiths,R.C. and Marjoram,P. (1997) An ancestral recombination graph. In Donnelly,P. and Tavaré,S. (eds), *Progress in Population Genetics and Human Evolution*. Springer-Verlag, Berlin, Vol. 87, pp. 257–270.

Hein,J. (1993) A heuristic method to reconstruct the history of sequences subject to recombination. *J. Mol. Evol.*, **36**, 396–405.

Holland,B.R. *et al.* (2002) Delta plots: a tool for analyzing phylogenetic distance data. *Mol. Biol. Evol.*, **19**, 2051–2059.

Husmeier,D. and McGuire,G. (2003) Detecting recombination in 4-taxa DNA sequence alignments with Bayesian hidden Markov models and Markov chain Monte Carlo. *Mol. Biol. Evol.*, **20**, 315–337.

Husmeier,D. and Wright,F. (2004) Detecting recombination in DNA sequence alignments. In Husmeier,D., Dybowski,R. and Roberts,S. (eds), *Probabilistic Modelling in Bioinformatics and Medical Informatics*. Springer.

Huson,D.H. (1998) SplitsTree: analyzing and visualizing evolutionary data. *Bioinformatics*, **14**, 68–73.

Jukes,T.H. and Cantor,C.R. (1969) Evolution of protein molecules. In Munro,H.N. (ed.), *Mammalian Protein Metabolism*, pp. 21–132.

Kececioglu,J.D. and Gusfield,D. (1998) Reconstructing a History of Recombinations From a Set of Sequences. *Discrete Appl. Math.*, **88**, 239–260.

Kimura,M. (1980) A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.*, **16**, 111–120.

Kuhner,M.K. *et al.* (2000) Maximum likelihood estimation of recombination rates from population data. *Genetics*, **156**, 1393–1401.

Lajoie,M. and El-Mabrouk,N. (2005) Recovering haplotype structure through recombination and gene conversion. *Bioinformatics*, **21** (Suppl. 2), ii173–ii179.

Lole,K.S. *et al.* (1999) Full-length human immunodeficiency virus type 1 genomes from subtype C-infected seroconverters in India, with evidence of intersubtype recombination. *J. Virol.*, **73**, 152–160.

McGuire,G. and Wright,F. (2000) TOPAL 2.0: improved detection of mosaic sequences within multiple alignments. *Bioinformatics*, **16**, 130–134.

Mevissen,H.T. and Vingron,M. (1996) Quantifying the local reliability of a sequence alignment. *Protein Eng.*, **9**, 127–132.

Negroni,M. and Buc,H. (2001) Mechanisms of retroviral recombination. *Annu. Rev. Genet.*, **35**, 275–302.

Posada,D. (2002) Evaluation of methods for detecting recombination from DNA sequences: empirical data. *Mol. Biol. Evol.*, **19**, 708–717.

Posada,D. and Crandall,K.A. (2001) Evaluation of methods for detecting recombination from DNA sequences: computer simulations. *Proc. Natl Acad. Sci. USA*, **98**, 13757–13762.

Rambaut,A. and Grassly,N.C. (1997) Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. *Comput. Appl. Biosci.*, **13**, 235–238.

Salminen,M. (2003) Detecting recombination in viral sequences. In Salemi,M. and Vandamme,A.-M. (eds), *The Phylogenetic Handbook: A Practical Approach to DNA and Protein Phylogeny*. Cambridge University Press.

Sawyer,S. (1989) Statistical tests for detecting gene conversion. *Mol. Biol. Evol.*, **6**, 526–538.

Schierup,M.H. and Hein,J. (2000) Consequences of recombination on traditional phylogenetic analysis. *Genetics*, **156**, 879–891.

Smith,J.M. (1992) Analyzing the mosaic structure of genes. *J. Mol. Evol.*, **34**, 126–129.

Song,Y.S. and Hein,J. (2005) Constructing minimal ancestral recombination graphs. *J. Comput. Biol.*, **12**, 147–169.

Song,Y.S. *et al.* (2005) Efficient computation of close lower and upper bounds on the minimum number of recombinations in biological sequence evolution. *Bioinformatics*, **21** (Suppl. 1), i413–i422.

Spang,R. *et al.* (2002) A novel approach to remote homology detection: jumping alignments. *J. Comput. Biol.*, **9**, 747–760.

Spencer,M. (2003) Exact significance levels for the maximum chi(2) method of detecting recombination. *Bioinformatics*, **19**, 1368–1370.

Weiller,G.F. (1998) Phylogenetic profiles: a graphical method for detecting genetic recombinations in homologous sequences. *Mol. Biol. Evol.*, **15**, 326–335.

Wiuf,C. *et al.* (2001) A simulation study of the reliability of recombination detection methods. *Mol. Biol. Evol.*, **18**, 1929–1939.

Yang,Z. (1996) Among-site rate variation and its impact on phylogenetic analyses. *Trends Ecol. Evol.*, **11**, 367–372.

Zimmer,R. and Lengauer,T. (1997) Fast and numerically stable parametric alignment of biosequences. In *Proceedings of the first annual international conference on Computational molecular biology RECOMB '97,* New York, NY, USA, ACM Press.