In search of lost introns

Miklós Csúrső, J. Andrew Holey and Igor B. Rogozin

1Department of Computer Science and Operations Research, Université de Montréal, Québec, Canada,
2Department of Computer Science, Saint John’s University and the College of St. Benedict, Collegeville, MN, USA
and 3National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health,
Bethesda, MD, USA

ABSTRACT
Many fundamental questions concerning the emergence and subsequent evolution of eukaryotic exon–intron organization are still unsettled. Genome-scale comparative studies, which can shed light on crucial aspects of eukaryotic evolution, require adequate computational tools.

We describe novel computational methods for studying spliceosomal intron evolution. Our goal is to give a reliable characterization of the dynamics of intron evolution. Our algorithmic innovations address the identification of orthologous introns, and the likelihood-based analysis of intron data. We discuss a compression method for the evaluation of the likelihood function, which is noteworthy for phylogenetic likelihood problems in general. We prove that after \( O(n/\log n) \) preprocessing time, subsequent evaluations take \( O(n/\log^2 n) \) time almost surely in the Yule–Harding random model of \( n \)-taxon phylogenies, where \( n \) is the input sequence length.

We illustrate the practicality of our methods by compiling and analyzing a data set involving 18 eukaryotes, which is more than in any other study to date. The study yields the surprising result that ancestral eukaryotes were fairly intron-rich. For example, the biterian ancestor is estimated to have had more than 90% as many introns as vertebrates do now.

Availability: The Java implementations of the algorithms are publicly available from the corresponding author’s site http://www.iro.umontreal.ca/~csuros/introns/. Data are available on request.
Contact: csuros@iro.umontreal.ca

1 INTRODUCTION
Typical eukaryotic protein-coding genes contain introns, which are removed prior to translation. Key constituents of the spliceosome, which is the RNA–protein complex that performs the intron excision, can be traced back (Collins and Penny, 2005) to the last common ancestor of extant eukaryotes (LECA). Even deep-branching lineages (Nixon et al., 2005) to the last common ancestor of extant eukaryotes, can be traced back (Collins and Penny, 2005). The assumptions used in these studies require adequate computational tools.

The usual procedure for identifying conserved portions of a multiple alignment. Section 3 describes a likelihood framework in which intron evolution can be analyzed in a theoretically sound manner. Section 4 scrutinizes a compression technique that accelerates the evaluation of the likelihood function. Fast evaluation is particularly important when the likelihood is maximized in a numerical procedure that computes the likelihood function with many different parameter settings. Section 5 describes two applications of our methods. In one application, intron-aware alignment was used to validate some unexpected features of intron evolution before LECA. In the second application, we analyzed intron evolution in 18 eukaryotic species. We found evidence of intron-rich early eukaryotes and a prevalence of intron loss over intron gain in recent times.

2 IDENTIFICATION OF ORTHOLOGOUS INTRONS
2.1 Intron-aware alignment
Orthologous introns can be identified by using whole genome alignments in the case of closely related genomes (Coulombe-Huntington and Majewski, 2007). For distant eukaryotes, however, intron orthology can only be established through protein alignments (Rogozin et al., 2005). The usual procedure is to project intron positions onto an alignment of multiple orthologous proteins. If introns in different species are projected to the same alignment position, then the introns are assumed to be orthologous.

Introns can also serve as checkpoints in a protein alignment. If the protein sequences are annotated with the intron
positions, then the alignment can exploit intron homology by rewarding aligned introns and penalizing the juxtaposition of an intron with an intronless codon. In addition to appropriate scoring, alignment algorithms need to be modified to cope with phase-0 introns, which fall between codons. Incorporating intron annotation should lead to better alignments at the amino acid level (Fig. 1), and to a more reliable identification of orthologous introns.

First, consider the case of aligning two protein sequences, \( S \) and \( T \), which are annotated with the intron positions. Every residue has two associated intron sites (after the first and second nucleotides of their codons), and there is an intron site between consecutive amino acid positions (phase-0 introns). Intron sites may or may not contain introns. We use the notation for \( S[i] \) for the phase-0 site preceding the codon for amino acid \( i \), and \( S[i+1] \) for phase-1 and phase-2 sites within the codon. Intron presence is encoded by 1, and intron absence is encoded by 0 throughout this article. The intron annotation is specified by the variables \( S[i,j] \in \{1,0\} \).

Phase-1 and phase-2 intron sites are automatically placed by their associated amino acids, but the placement of phase-0 introns is not fixed with respect to gaps. It is not possible to simply add a new character representing phase-0 introns because they affect gaps differently from amino acids. Scores for aligned intron sites are given by a \( 2 \times 2 \) matrix \( \Lambda \). If \( M \) is the amino acid scoring matrix, then the alignment of \( S[i] \) and \( T[j] \) entails a score of \( M S[i] T[j] + \Lambda S[i] T[j] \) with an indel.

The intron scores \( \Lambda \) can be based on log-likelihood ratios in a probabilistic model (Durbin et al., 1998). For an aligned intron site, let \( s, t \in \{1,0\} \) denote the intron states in \( S \) and \( T \), respectively. The score \( \Lambda \) should be proportional to 

\[
\log \left( \frac{\pi_S(s)\pi_T(t)}{p(s,t)} \right),
\]

where \( p \) denotes the joint distribution of states at orthologous sites, and \( \pi_S, \pi_T \) are the prior intron state probabilities.

In order to assess the strength of intron-match signals, and to choose reasonable scores, we used the data set of Rogozin et al. (2003). We estimated \( p \) and \( \pi \) using relative frequencies of intron absence-presence patterns in the 488 157 possible intron sites, and then computed log-odds scores for intron matches. The intron score is asymmetric and varies with evolutionary distance and intron conservation. Intron-less sites have an insignificant score, but shared introns have a high score, such as 304 (human–Plasmodium), 317 (human–Arabidopsis) or 515 (Drosophila–Anopheles) on a 1/60-bit scale. Shared introns thus give a signal comparable to strong amino acid matches: in the 1/60-bit scaled version of the VTML240 matrix (Müller et al., 2002), a tryptophan match scores 289. Discordant intron sites have similar scores to amino acid mismatches.

We added intron scoring into a multiple alignment framework, using a sum-of-pairs scoring policy with affine gap-scoring. It is NP-hard to find the alignment of two multiple alignments under these optimization criteria (Ma et al., 2003). Even the alignment of a single sequence to a multiple alignment necessitates sophisticated techniques (Kececioglu and Zhang, 1998). Our solution therefore uses a gap-counting heuristic: namely, a gap-open penalty is triggered for an indel aligned with an amino acid if the indel is preceded by an amino acid or a phase-0 intron. Gap opening thus corresponds to a pattern \( ?x \) or \( ?x_0 \). Here, 1, 0 are intron states for the phase-0 site, and \( ? \) denotes either state. In addition, \( x \) denotes an arbitrary amino acid, \( - \) is the indel character, and \( * \) is either of the latter two. We implemented affine gap-scoring by separate gap-open and -close penalties, so that gaps at the alignment extremities can be penalized less severely. Gap closing is counted for the patterns \( x_0 \) and \( -0x \).

Table 1 gives the recurrences for a dynamic programming algorithm that aligns an intron-annotated sequence \( S \) to a multiple alignment \( P \) of \( h \) intron-annotated protein sequences. In order to simplify the presentation, we represent the sequences in such a way that every odd position of \( S \) and \( P \) is a regular residue or alignment column, annotated with information on the presence of phase-1 and phase-2 introns, whereas every even position is a phase-0 intron site. We use \( \Lambda S[i] P[j] \) to denote the sum of intron-match scores for the intron sites associated with the positions \( S[i], P[j] \). We use also the shorthand \( M x y \) to denote scoring for the alignment of an amino acid \( x \) with an amino acid profile \( y \). The algorithm uses three types of variables, \( A[i,j], gS[i,j] \) and \( gP[i,j] \), which correspond to partial prefix alignments ending with aligned residues, gaps in \( S \), or gaps in \( P \), respectively. In case of \( gS[i,j] \), the last indel must be aligned with an amino acid column, and, thus \( j \) must be odd; for \( gP[i,j] \), \( i \) must be odd.

Gaps are scored by using affine penalties, with gap-open, -extend, and -close scores, denoted by \( \gamma^- \), \( \gamma^- \), \( \gamma^- \). The gap-counting heuristic implies that gap scores in the equations of Table 1 are defined by the number of certain patterns in up to three consecutive alignment columns. Table 2 lists the patterns for the gap-counting heuristic. Table 1 does not show the initialization of the variables, nor the final gap-counting: they employ a logic analogous to the recurrences. At the end of the algorithm, the best of \( A[S][P] \), \( gS[S][P] \), \( gP[S][P] \) is selected, and the actual alignment is found by standard traceback techniques (Durbin et al., 1998).

We implemented the algorithm in Java. The program iteratively realigns one sequence at a time to the rest of the sequences in a multiple alignment. Instead of
### Table 1. Recurrences for intron-aware alignment

| Recurrence | Condition | Formula |
|------------|-----------|---------|
| $A[i, j] = M^{\frac{S(i)}{P(j)}} + A^{\frac{S(j)}{P(j)}} + y_1^{(c)}(j) + \max\{A[i - 2, j - 2] + A^{\frac{S(i - 1)}{P(j - 1)}} + y_1^{(c)}(j) + y_1^{(c)}(j), odd \ i, odd \ j\}$ |
| $gS[i - 2, j - 2] + A^{\frac{S(i - 1)}{P(j - 1)}} + y_1^{(c)}(j) + y_2^{(2)}(j - 2), gS[i - 1, j - 2] + A^{\frac{0}{P(j - 1)}} + y_1^{(c)}(j) + y_2^{(2)}(j - 2)$ |
| $gP[i - 2, j - 2] + A^{\frac{S(i - 1)}{P(j - 1)}} + y_2^{(c)}(j) + y_3^{(3)}(j), gP[i - 2, j - 1] + A^{\frac{S(i - 1)}{0^0}} + y_3^{(3)}(j) + y_2^{(2)}(j)$ |
| $gS[i, j] = A^{\frac{0}{P(j)}} + y_2^{(3)}(j) + \max\{A[i - 2, j - 2] + A^{\frac{0}{P(j - 1)}} + y_3^{(c)}(j) + y_1^{(c)}(j), odd \ i, odd \ j\}$ |
| $gS[i, j - 2] + A^{\frac{0}{P(j - 1)}} + y_2^{(c)}(j) + y_3^{(3)}(j), gS[i - 1, j] + A^{\frac{0}{0^0}} + y_3^{(3)}(j) + y_2^{(2)}(j)$ |
| $gP[i, j - 2] + A^{\frac{S(i)}{0^0}} + y_1^{(c)}(j) + \max\{A[i - 2, j - 2] + A^{\frac{S(i - 1)}{0^0}} + y_3^{(3)}(j) + y_2^{(2)}(j), odd \ i, odd \ j\}$ |
| $gS[i - 2, j] + A^{\frac{S(i - 1)}{0^0}} + y_2^{(3)}(j) + y_3^{(3)}(j), gS[i - 1, j] + A^{\frac{0}{0^0}} + y_3^{(3)}(j) + y_2^{(2)}(j)$ |
| $gP[i - 2, j] + A^{\frac{S(i - 1)}{0^0}} + y_3^{(3)}(j) + y_2^{(2)}(j)$ |
| $gS[i, j] = A^{\frac{0}{P(j)}} + y_2^{(3)}(j) + \max\{A[i - 1, j - 2] + A^{\frac{S(i)}{P(j - 1)}} + y_3^{(c)}(j) + y_1^{(3)}(j), even \ i, odd \ j\}$ |
| $gS[i - 1, j - 2] + A^{\frac{S(i)}{P(j - 1)}} + y_1^{(c)}(i, j) + y_2^{(2)}(i, j), gS[i - 1, j - 2] + A^{\frac{0}{P(j - 1)}}$ |
| $gP[i - 1, j - 2] + A^{\frac{S(i)}{P(j - 1)}} + y_1^{(c)}(j) + y_3^{(3)}(j), gP[i - 1, j - 1] + A^{\frac{S(i)}{0^0}} + y_3^{(3)}(j) + y_2^{(2)}(j)$ |
| $gP[i, j] = A^{\frac{S(i)}{0^0}} + y_1^{(c)}(j) + \max\{A[i - 2, j - 1] + A^{\frac{S(i - 1)}{P(j)}} + y_3^{(c)}(j) + y_3^{(3)}(j), odd \ i, even \ j\}$ |
| $gS[i - 2, j - 1] + A^{\frac{S(i - 1)}{P(j)}} + y_2^{(3)}(j) + y_3^{(3)}(j - 1), gS[i - 1, j - 1] + A^{\frac{0}{P(j)}} + y_3^{(3)}(j) + y_2^{(2)}(j - 1)$ |
| $gP[i - 2, j - 1] + A^{\frac{S(i - 1)}{P(j)}} + y_2^{(3)}(j) + y_3^{(3)}(j), gP[i - 2, j] + A^{\frac{S(i - 1)}{0^0}}$ |

Odd positions correspond to regular amino acids and possibly phase-1 and -2 introns. Even positions are placeholders for phase-0 intron sites.
Table 2. Patterns for gap counting

| Pattern | Definition | Example |
|---------|------------|---------|
| \(y^{(0)}\) | \(1-\text{or }x0\) | \(y^{(0)}\) |
| \(y^{(1)}\) | \(x\) | \(y^{(1)}\) |
| \(y^{(2)}\) | \(1\text{ or }x0\) | \(y^{(2)}\) |
| \(y^{(3)}\) | \(x\) | \(y^{(3)}\) |
| \(y^{(4)}\) | \(-x0\text{ or }-1\text{x}\) | \(y^{(4)}\) |
| \(y^{(5)}\) | \(x\) | \(y^{(5)}\) |
| \(y^{(6)}\) | \(-1\) | \(y^{(6)}\) |
| \(y^{(7)}\) | \(-\) | \(y^{(7)}\) |

For each penalty type, the number of rows with the given pattern equals the number of gaps opened \(y^{(1)}\), closed \(y^{(2)}\) or extended \(y^{(3)}\). Pattern symbols: phase-0 intron states 1, 0, arbitrary amino acid \(x\), indel --, and * is either of the latter two. The pattern’s rightmost symbol is in column \(j\). For instance, \(y^{(3)}(i,j)\) equals the gap-open penalty multiplied by the number of such rows in profile \(P\) where column \(j\) contains an indel, and column \(j-1\) has a phase-0 intron. \(y^{(4)}(i,j)\) and \(y^{(5)}(i,j)\) equal 0 if if \(S[j]\) contains no intron.

Sequence-dependent intron match-mismatch scores, the implementation uses only two parameters: one for intron conservation and another for intron-state mismatch.

2.2 Identification of conserved blocks

In order to reliably identify orthologs, we need to be able to distinguish regions of the alignment that are highly conserved from those that are less conserved. In poorly conserved regions, we cannot confidently infer intron orthology.

Zhang et al. (1999) proposed postprocessing pairwise sequence alignments into alternating blocks that score above a threshold parameter \(\alpha\) or below \((-\alpha\). We attained a similar goal by adapting algorithmic techniques from Cs"uros (2004). The procedure separates a multiple alignment into alternating high- and low-scoring regions. Using a complexity penalty \(\alpha\), a segmentation with \(k\) high-scoring regions has a segmentation score of \(A-\alpha k\) where \(A\) is the sum of scores for the aligned columns. Column scores are computed without gap-open and -close penalties. The best segmentation of an alignment of length \(\ell\) can be found in \(O(\ell)\) time after the column scores are computed.

The result of this computation is that the total of column scores in each selected high-scoring region will be greater than \(\alpha\). There may be small subregions of negative scores, but the total score of such a subregion cannot be less than \((-\alpha\). Conversely, unselected regions score below \((-\alpha)\) and cannot have subregions scoring above \(\alpha\).

3 A LIKELIHOOD FRAMEWORK

3.1 Markov models of evolution

We use a Markov model for intron evolution, as in previous studies (Cs"uros, 2005). For the sake of generality, we describe the Markov model (Felsenstein, 2004; Steel, 1994) over an arbitrary alphabet \(A\) of fixed size \(r=|A|\). The intron alphabet is \(A=[0,1]\). A phylogeny over a set of species \(X\) is defined by a rooted tree \(T\) and a probabilistic model. The leaves are bijectively mapped to elements of \(X\). Each tree node \(u\) has an associated random variable \(\xi(u)\), which is called its state or label, that takes values over \(A\). The tree \(T\) with its parameters defines the joint distribution for the random variables \(\xi(u)\).

Along every path away from the root, the node states form a Markov chain. The distribution is thus determined by the root probabilities \(\pi(a):a\in A\) and edge transition probabilities \((p_{a}(a\rightarrow b):a,b\in A)\) assigned to every edge \(e\). The root probabilities give the distribution of the root state. Edge transition probabilities define the conditional distributions \(P[\xi(t)=b|\xi(u)=a]=p_{a}(a\rightarrow b)\). The leaf states form the character \(\xi=(\xi(u):u\in X)\). An input data set (or sample) consists of independent and identically distributed (iid) characters: \(\xi=(\xi_{i}:i=1,\ldots,\ell)\).

In case of intron evolution, introns are generated by a two-state continuous-time Markov process with gain and loss rates \(\kappa_{G},\kappa_{L}\geq 0\) along each edge \(e\). The edge length is denoted by \(t_{e}\).

Using standard results, the transition probabilities on edge \(e\) with rates \(\kappa_{G}=\lambda,\kappa_{L}=\mu\) and length \(t_{e}\) can be written as \(p_{0}(0\rightarrow 1)=\lambda/(\lambda+\mu)(1-\exp(-\lambda t_{e}))\), \(p_{1}(1\rightarrow 0)=\mu/(\lambda+\mu)(1-\exp(-\mu t_{e}))\), and \(p_{0}(0\rightarrow 0)=1-p_{0}(0\rightarrow 1),p_{1}(1\rightarrow 1)=1-p_{1}(1\rightarrow 0)\).

Independent model parameters are thus \(\pi(1),\mu_{G},\mu_{L}\) for all edges \(e\). It is important to allow for branch-dependent rates, since loss and gain rates vary considerably between lineages (Jeffares et al., 2006; Roy and Gilbert, 2006).

In a maximum likelihood approach, model parameters are set by maximizing the likelihood of the input sample. Let \(x=(x_{1,\ldots,x_{j}})\) be the input data. Each \(x_{i}\) is a vector of \(n\) states, and we write \(x(u)\) to denote the observed state of leaf \(u\). By independence, the likelihood of \(x\) is the product \(P[\xi=x]=\prod_{u}[P[\xi(u)=x_{u}]]\). Each character’s likelihood can be computed in \(O(n)\) time, using a dynamic programming procedure introduced by Felsenstein (1981). The procedure relies on a ‘pruning’ technique, which consists of computing the conditional likelihoods \(L_{u}(a)\) for every node \(u\) and letter \(a\). \(L_{u}(a)\) is the probability of observing the leaf labelings in the subtree of \(u\), given that \(u\) is labeled with \(a\). The likelihood for the character \(x\) equals \(L(x)=P[\xi=x]=\sum_{a}\pi(a)L_{\text{root}}(a)\).

Intron data are somewhat unusual in that an all-0 character is never observed: the input does not include sites in which introns are absent in all of the organisms. To resolve this difficulty, we employ a correction technique proposed by Felsenstein (1992) for restriction sites. (Cs"uros, 2005) describes an alternative technique based on expectation maximization.)

We compute the likelihood under the condition that the input does not include all-0 characters. We use therefore the corrected likelihood \(L'(x)=L(x)/(1-L(0^{n}))\) and maximize \(L'=\prod_{i}L'(x_{i})\).

3.2 Ancestral events in intron evolution

Our goal is to give a reliable characterization of the dynamics of intron evolution. In particular, we aim to give estimates for intron density in ancestral species, and for intron loss and gain events on the edges. Notice that the estimation method needs to account for ancestral introns even if they got eliminated in all descendant lineages. It is possible to do that with the help of conditional expectations, which fit naturally into a likelihood framework.

For an observed character \(x\), we define upper conditional likelihoods \(U_{x}(a)\) so that \(U_{x}(a)L_{x}(a)=P[\xi=x,\\xi(u)=a]\).
Upper conditional likelihoods are computed with dynamic programming, from the root towards the leaves, in \( O(n\ell) \) time (Csírsz, 2005), even for non-binary trees. Similar computations are routinely used in DNA and protein likelihood maximization programs (Adachi and Hasegawa, 1995; Guindon and Gascuel, 2003). Here we allow irreversible probabilistic models, which explains why \( U_u(a) \) must be computed in a top-down fashion in (1).

The posterior probability for the state at node \( u \) is
\[
q_{u}^{(a)}(x) = \mathbb{P}\{\xi(u) = a \mid \xi = x\} = \frac{U_u(a)L_u(a)}{\sum_{b \in A} U_u(b)L_u(b)}.
\]

The posterior probabilities for state changes on an edge \( uv \) are
\[
q_{uv}^{(a \to b)}(x) = \mathbb{P}\{\xi(u) = a, \xi(v) = b \mid \xi = x\} = \frac{U_u(a)L_u(a)p_{uv}(a \to b)L_v(b)}{\sum_{a' \in A} p_{uv}(a \to a')L_v(a')}
\]

Now, the number of ancestral introns is estimated as the conditional expectation \( N_u(x) = \sum_{a \in A} q_{u}^{(a)}(x) \). The formula takes into consideration unobserved intron sites, by estimating their number \( \ell = f((L(0^n))/(1 - L(0^n))) \) as the mean of a negative binomial random variable. The number of intron state changes is estimated as \( N_{uv}(x) = \ell q_{uv}^{(a \to b)}(x) \). In particular, \( N_u(1 \to 0) \) is the number of introns lost, and \( N_u(0 \to 1) \) is the number of introns gained on the edge leading to \( v \).

In order to compute \( U_u \), we initialize \( U_{\text{root}}(a) = \pi(a) \). On every edge \( uv \),
\[
U_v(b) = \sum_{a \in A} U_u(a)p_{uv}(a \to b) \prod_{y \in \text{siblings}(v)} \sum_{a' \in A} p_{uv}(a \to a')L_v(a') \]
\[
= \sum_{a \in A} U_u(a)L_u(a)p_{uv}(a \to b)L_v(b) \prod_{a' \in A} p_{uv}(a \to a')L_v(a'). \tag{1}
\]

### 4.1 Yule–Harding model

The Yule–Harding distribution is encountered in random birth and death models of species and in coalescent models (Felsenstein, 2004), and is thus one of the most adequate random models for phylogenies. In one of the equivalent formulations, a random tree is grown by adding leaves one by one in a random order. The leaves are first numbered by using a random uniform permutation of the integers \( 1, 2, \ldots, n \). Leaves are joined to the tree in an iterative procedure. In step 1, the tree is just leaf 1 on its own. In step 2, the tree is a ‘cherry’ with leaves 1 and 2. In each subsequent step \( i = 3, \ldots, n \), a random leaf \( Y_i \) is picked uniformly from the set \( \{1, 2, \ldots, i-1\} \). The new leaf \( i \) is added to the tree as the sibling of \( Y_p \), forming a cherry: a new node is placed on the edge leading to \( Y_i \) and \( i \) is connected to it. The resulting random tree in iteration \( n \) has the Yule–Harding distribution (Harding, 1971).

### 4.2 Preprocessing

The likelihood can be computed faster by first identifying the different \( x_i \) values, along with their multiplicity in the data. For large trees, the input typically consists of many different labelings, but for small trees with \( n > \log \ell \), the compression is useful, since the number of different \( x_i \) values is bounded by \( r^\ell < \ell \). In order to exploit the benefits of compression, we extend it to every subtree.

**Definition 1.** Define the multisets of observed labelings for every node \( u \) as follows. Let \( n' \) denote the number of leaves in the subtree rooted at \( u \), and let \( a_1, \ldots, a_{n'} \) denote those leaves. Define \( v_u(y) \) for every labeling \( y \in A^n \) of the leaves in the subtree of \( u \) as the number of times \( y \) is observed in the data:
\[
v_u(y) = \sum_{i=1}^{n'} \chi\{y_k = x_i(u_k)\},
\]
where \( X\{\cdot\} \) denotes the indicator function that takes the value 1 if its argument is true, otherwise it is 0. Define also the set of observed labelings
\[
S_u = \{y \in A^n : v_u(y) > 0\}.
\]
Algorithm COMPRESS
C1 for every leaf u do
C2 initialize α[1..r] ← 0 // α[j] is the first occurrence of state j
C3 for i = 1, ..., ℓ do
C4 if α[x(u)] = 0 then α[x(u)] ← i
C5 set h0(u) ← a[x(u)]
C6 for every non-leaf node u in a post-order traversal do
C7 initialize the map H: {1, ..., ℓ}2 → {1, ..., ℓ} as empty
C8 let v, w be the left and right children of u
C9 for i = 1, ..., ℓ do
C10 if H(hi(v), hi(w)) = null then H(hi(v), hi(w)) ← i
C11 set hi(u) ← H(hi(v), hi(w))

Fig. 2. Computing the auxiliary arrays from which the multiplicities vi are obtained.

The multisets of observed labelings are computed in the preprocessing step. The likelihood function is evaluated subsequently by computing the conditional likelihoods at each node u for the labelings of S_u only, in O(|S_u|) time.

In order to compute vi, we use a recursive procedure. It is important to avoid working with the O(n)-dimensional vectors y of Definition 1 directly, otherwise the preprocessing may take superlinear time in n. For that reason, every labeling y ∈ S_u is represented by the index i for which x_i is the first occurrence of yi. Accordingly, we compute an auxiliary array h(u), which stores the first occurrence of each labeling x_i in u’s subtree. In particular, h(u) = i if i is the smallest index such that x_i(u_k) = x_i(u_k) for all k where u_k are the leaves in u’s subtree. Figure 2 shows that the values h(u) can be computed in a postorder traversal. After h(u) are computed for all i and u, the multiplicities vi and observed labelings S_u are straightforward to calculate in linear time. The map H in Lines C7–C11 is sparse with at most ℓ entries, and can be implemented as a hash table so that accessing and updating it takes O(1) time. Consequently, Algorithm COMPRESS takes O(nℓ) time.

4.3 Evaluating the likelihood function
After the preprocessing step, the conditional likelihoods are computed only for the different labelings within each subtree. Figure 3 shows the evaluation of the likelihood function. The running time for the algorithm is O(s) where s is the total number of different labelings within all subtrees: s = ∑ u |S_u|. If n_u denotes the number of leaves in the subtree rooted at u, then S_u has at most n_u elements. Hence, s is bounded as

\[ s \leq \sum u \min(n_u, \ell). \]  

(2)

Observe that by sheer number of arithmetic operations, it is always worth evaluating the likelihood function this way. The worst situation is that of a caterpillar tree (where every inner node has a leaf child). In that case, there are only a few (log_2 ℓ - 1) non-leaf nodes for which we can compress the data, and it is possible to construct an artificial data set in which there are ℓ different labelings for n = O(log ℓ) subtrees.

Algorithm LOGLIKELY
L1 for every leaf u do
L2 set L_u[a] = a' ← \( \chi(a = a') \) for all a ∈ S_u
L3 for every non-leaf node u in a post-order traversal do
L4 for every labeling y ∈ S_u do
L5 let u_1, u_2 be the children of u, and let y_1, y_2 be their subtree labelings
L6 for all a ∈ A
L7 compute \( L_u[a] \) = \( \prod_{j=1}^{1,2} \sum_{a' \in A} \mu_{uj}(a \rightarrow a') L_{u_j}[y_j][a'] \)
L8 set logL ← 0
L9 for every labeling y ∈ S_root do
L10 logL ← logL + \mu_root(y) \cdot \log_{a \in A} \prod (\pi(a) L_{root}[y][a])
L11 return logL.

Fig. 3. Computing the log-likelihood using the observed labelings.

Caterpillar trees, however, are rare in phylogenetic analysis. Typical phylogenies have fairly balanced subtrees (Aldous, 2001).

In what follows, we examine the bound of (2) more closely in the Yule–Harding model. The analysis relies on a characterization of the random number of subtrees with a given size, as expressed in Theorems 1 and 2.

**Theorem 1.** Consider random evolutionary trees with n leaves in the Yule–Harding model. Let C_k denote the number of subtrees with k = 1, ..., n - 1 leaves in a random tree. The expected value of C_k is

\[ \mathbb{E}[C_k] = 2n \left( \frac{1}{k} - \frac{1}{k + 1} \right). \]  

(3)

**Trivially, \( \mathbb{E}[C_n] = 1 \):**

**Proof.** Heard (1992) derives Equation (3) by appealing to a Polya urn model. An equivalent result is stated by Devroye (1991).

**Theorem 2.** For all \( \epsilon > 0 \),

\[ \mathbb{P}[C_k \leq \mathbb{E}[C_k] - \epsilon] \leq e^{-2\epsilon^2}; \]  

(4a)

\[ \mathbb{P}[C_k \geq \mathbb{E}[C_k] + \epsilon] \leq e^{-2\epsilon^2}. \]  

(4b)

**Proof.** Consider the random construction of the tree. Let \( Y_i \) denote the random leaf picked in step i to which leaf i gets connected, for \( i = 3, 4, \ldots, n \). Each random variable \( Y_i \) is uniformly distributed over the set \{1, 2, ..., i - 1\}, and \( Y_3, Y_4, \ldots, Y_n \) are independent. Moreover, they completely determine the tree \( T \) at the end of the procedure. Consequently, \( C_k \) is a function of \( \{Y_i : i = 3, \ldots, n\} \): \( C_k = f(Y_3, Y_4, \ldots, Y_n) \). The key observation for the concentration result is that if we change the value of only one of the \( Y_i \) in the series, then \( C_k \) changes by at most two. In order to see this, consider what happens to the tree \( T \), if we change the value of exactly one of the \( Y_i \) from y to y'. Such a change corresponds to a ‘subtree prune and regraft’ transformation (Felsenstein, 2004). Specifically, the subtree \( T_y \) defined as the child tree of the lowest common ancestor \( u \) of y
and $i$ containing the leaf $i$, is cut from $T$, and is reattached to one of the edges on the path from the root to $y'$. Now, such a transformation does not affect $C_k$ by much. Notice that subtree sizes are strictly monotone decreasing from the root on every path. On the path from the root to $u$, subtree sizes decrease by the size $t$ of $T$, and $u$ disappears, contributing a change of $+1, 0$ or $-1$ to $C_k$. (At most one subtree of size $t+k$ that contains $T$, now has size $k$, and at most one subtree of original size $k$ is not counted anymore: it may be $u$’s subtree itself, or a subtree above it.) An analogous argument shows that regrafting contributes a change of $+1, 0$, or $-1$. Hence, the function $f(\cdot)$ is such that by changing one of its arguments, its value changes by at most 2. As a consequence, McDiarmid’s inequality (1989) can be applied to bound the probabilities of large deviations for $C_k$. In particular, for all $\epsilon > 0$, \[
P\{f(Y_1, \ldots, Y_n) - \mathbb{E}f(Y_1, \ldots, Y_n) \leq -\epsilon\} \leq e^{-2\epsilon^2/c^2}\]
where $c^2 = \sum_{j=3}^t c_j^2$ and 
\[c_j = \max_{y_1, \ldots, y_j, y'j} \left| f(y_1, \ldots, y_{j-1}, y, y_{j+1}, \ldots, y_n) - f(y_1, \ldots, y_{j-1}, y', y_{j+1}, \ldots, y_n) \right|.
\]
Since $c_2 = 2$ for all $i$, $P\{C_k \leq \mathbb{E}C_k - \epsilon\} \leq e^{-\epsilon^2/(2n-2\text{log} n)}$, implying Equation (4a). An identical bound holds for the right-hand tail of the distribution. \hfill \square

**Remark.** The particular case of $k = 2$ was considered by McKenzie and Steel (2000). They showed that the distribution of $C_2$ is asymptotically normal with mean $n/3$ and variance $2n/45$. The result suggests that the best constant factor in the exponent of Equation (4) is $45/8$ for $k = 2$, instead of $1/2$. Rosenberg (2006) gave exact formulas for the variance of $C_k$. He showed that the variance of $C_k$ is $\nu_k/n^2$ where $8/45 \leq \nu_k < 2$ is a constant that depends on $k$. The variance formulas were given earlier in a different context by Devroye (1991). (See also the discussion by Blum and François (2005).) The result suggests that by analogy with the cherries, the best constant factor in the exponent is $k^2/n^2$. It is thus plausible that the probability is properly bounded by $1-o(n \text{log}^2 \ell)$ in Theorem 3.

**Theorem 3.** With probability $1-o(n/(\text{log}^4 \ell))$, the likelihood function can be evaluated for random trees in the Yule–Harding model in $O(n \text{log} \ell)$ time after an initial preprocessing step that takes $O(n \text{log} \ell)$ time. Evaluating the likelihood function takes $O(n \text{log} \ell)$ time in the worst case, and $O(n \text{log}^4 \ell)$ time on average.

We need the following lemma for the proof of Theorem 3.

**Lemma 4.** For all $t \geq 4$, $\sum_{k=1}^{t} \frac{t^{k-1}}{k(k+1)} \leq \frac{2^t}{t+1}$. For all $t \geq 1$ and $r = 3, 4, \ldots, \sum_{k=1}^{t} \frac{t^{k-1}}{k(k+1)} \leq \frac{2^t}{t+1}$.

**Proof.** The proof is straightforward by induction in $t$. Notice that the right order of magnitude is $\sum_{k=1}^{t} (t^k/k(k+1)) = \Theta(t^2/2)$ but we need a bound for all $t$.

**Proof of Theorem 3.** The preprocessing (Fig. 2) takes $O(n \text{log} \ell)$ time as discussed in Section 4.2. The evaluation of the likelihood function (Fig. 3) takes $O(s)$ time. By Equation (2),
\[
s \leq \sum_{k=1}^{n} C_k \text{min}\{r^k, \ell\} = \sum_{k=1}^{[\text{log} \ell]} C_k r^k + \sum_{k=1+\text{[log} \ell]}^{n} C_k \ell.
\]

Let $t = [\text{log}_s \ell]$. By Theorem 1 and Lemma 4, if $\ell \geq 16$ or $r \geq 3$,
\[
\mathbb{E}r \leq 2n \sum_{k=1}^{t} \frac{r^k}{t(k+1)} + \ell \left(\frac{2n}{t+1} - 1\right) \leq 4n \ell + \frac{1}{t+1},
\]
which proves our claim about the average running time. Now, let $\epsilon = \frac{t}{2n \ell t}$ Plugging $\epsilon$ into Theorem 2, we get that
\[
P\left\{\left| C_k - \mathbb{E}C_k \right| \geq \epsilon \right\} \leq 2\exp\left(-\frac{n}{8} \left(\frac{1}{t+1} - \frac{1}{t+1} \right)^2\right),
\]
where $\mathbb{E}C_k$ denotes the event that $|C_k - \mathbb{E}C_k| < \epsilon$ for $k = 1, 2, \ldots, t$, and $\mathbb{E}C_{t+1}$ denote the event that $\sum_{k=1}^{t} |C_k - \mathbb{E}C_k| < \epsilon$. Since $\sum C_k = 2n - 1$, $\Gamma_{t+1} = \mathbb{E}C_{t+1}$ implies $\mathbb{E}C_{t+1}$. By (7),
\[
P\left\{\left| C_k - \mathbb{E}C_k \right| \geq \epsilon \right\} \leq 2\exp\left(-\frac{n}{8} \left(\frac{1}{t+1} - \frac{1}{t+1} \right)^2\right),
\]
where $\Gamma_k$ denotes the complementary event to $\mathbb{E}C_k$. Now, $\Gamma_{t+1}$ also implies that $s \leq (5n\ell)/(t+1)$. Since the likelihood computation takes $O(s)$ time, the theorem holds. \hfill \square

Theorem 3 underestimates the actual speedups that the compression method brings about. Table 3 shows that compression results in a 50–500-fold speedup of the likelihood evaluation in practice. Notice also that the constants hidden behind the asymptotic notation are quite different between the preprocessing and evaluation steps: costly floating point operations are avoided in the preprocessing step. It is important to stress that the theoretical analysis does not rely on similarities in the input data: Theorem 3 and the bound of (2) hold for any sample of size $\ell$. Real-life data are expected to behave even better, as Table 3 illustrates.

Even though the theorem holds for arbitrary alphabets, the compression is less effective for large alphabets (amino acids for example) in practice. For DNA sequences, however, it should still be valuable: we conjecture that compression would accelerate likelihood optimization by at least an order of magnitude.

We implemented LogLikelihood, along with likelihood maximization and the posterior calculations of Section 3.2 in a Java package. Likelihood is maximized by setting loss and gain parameters for the edges, by using mostly the Broyden–Fletcher–Goldfarb–Shanno method (Press et al., 1997)

**Table 3.** Effect of the compression on three data sets

| $n$ | $\ell$ | $r$ | $n \ell t$ | $n \ell s_{\text{cool}}$ | bound | $s$ |
|-----|-----|-----|---------|------------------|--------|-----|
| 8  | 7236| 2  | 101304 | 1386 368 183 |
| 18 | 8044| 2  | 273496 | 19142 16 764 1196 |
| 47 | 5216| 3  | 479872 | 309120 65 743 10 305 |

The fourth column quantifies the direct evaluation method, the fifth column quantifies the effect of the compression restricted to the root, the sixth column corresponds to the bound of Equation (2) and the seventh column gives the exact value of $s$. The first data set is from Rogozin et al. (2003), the second data set is the one analyzed here in Section 5.2, and the third one is an unpublished data set we have worked on, where an ambiguity character is included in the alphabet.
whenever possible, and occasionally Brent’s line minimization (Press et al., 1997) for each parameter separately. It is noteworthy that the algorithmic improvements reduced the likelihood optimization time to a few minutes on a common desktop computer for the data set of Section 5.2, and made the execution of a large number of simulations possible.

5 APPLICATIONS

5.1 Ancient paralogs

In our first example, intron-aware alignment was used to reject the hypothesis of misleading protein alignments.

We used intron-aware alignment in a study about ancient eukaryotic paralogs (Sverdlov et al., 2007). In the study, 157 homologous gene families were examined across six species (Arabidopsis thaliana, Homo sapiens, Canorhabditis elegans, Drosophila melanogaster, Saccharomyces cerevisiae and S. pombe). These families were notable because they contained paralogous members in multiple eukaryotic species, but not in prokaryotes, and, thus, presumably underwent duplication in the lineage leading to LECA. Ancient paralogs within and across species share very few (in the order of a few percentages) introns. The finding is quite surprising, as recent paralogs, resulting from lineage-specific duplications, and also orthologs between human and Arabidopsis agree much more in their intron sites (Rogozin et al., 2003). Consequently, ancient paralogs either lacked introns at the time of their duplication, or their duplication involved removal of introns.

In one of the data validation steps for the study, we used intron-aware alignment with very high intron match rewards. With larger rewards for intron matches, more introns line up in the alignment, but the protein alignment gets worse. Even by using huge intron match rewards at the expense of corrupted protein alignments, we were not able to achieve intron sharing levels similar to that of human-Arabidopsis orthologs. The lack of intron agreement is therefore not an artifact of the protein alignments.

5.2 Intron-rich ancestors

We compiled a data set with 18 eukaryotic species to give a comprehensive picture of spliceosomal evolution among Eukaryotes.

**Data preparation** Genbank flatfiles and protein sequences were downloaded from RefSeq (Pruitt et al., 2007) and Ensembl (Hubbard et al., 2007). Exon-intron annotation was extracted from the Genbank flatfiles. The C. briggsae protein sequences and genome annotation were downloaded from WormBase (Bieri et al., 2007), and intron annotation was extracted from the GFF annotation file. Table 4 lists the data sources and the species abbreviations. We used the 684 ortholog sets, each corresponding to a cluster of orthologous groups, or KOG (Tatusov et al., 2003), from the study of Rogozin et al. (2003) as ‘seeds’ for compiling a set of putative orthologs for our species. For each seed (consisting of homologous protein sequences for eight species), we performed a position-specific iterated BLAST search (Altschul et al., 1997). In case of Plasmodium species, we used three iterations against a database of all protozoan peptide sequences in RefSeq. We used an E-value cutoff of $10^{-15}$ for retaining candidates. Each candidate hit was then used as a query in a reversed position-specific BLAST (rpsblast) search (Marchler-Bauer et al., 2007) against the KOG database. Candidates were retained after this point if they had the highest scoring hit (by rpsblast) against the same KOG as the KOG of the seed data, and they scored within 80% of the best such hit for the species.

From the resulting set of paralogs, we selected a putative ortholog set in the following manner. For each KOG, we selected all possible triples of human–Arabidopsis–Saccharomyces paralogs and kept the triple that had the highest score in alignments built by MUSCLE (Edgar, 2004). Alignment score was computed using the VTML240 matrix (Müller et al., 2002). Additional putative orthologs were added for one species at a time, by aligning each paralog individually to the current profile, and keeping the one that gave the largest alignment score. At this iterative addition, scoring was done with the VTML240 matrix, by summing the five-highest-pairwise scores between a candidate and already included sequences.

The resulting sets were then realigned using MUSCLE, and then realigned again using our intron-aware alignment with a gap penalty of 300, gap-extend penalty of 11, VTML240 amino acid scoring, intron-match scores of 300 and intron-mismatch penalties of 20. (The latter were established using different intron-match and -mismatch scores, and selecting the ones that gave the fewest number of intron sites while decreasing the score of the implied protein alignment by less than 0.1%.) Conserved portions were extracted using our segmentation program with a complexity penalty of $\alpha = 400$ (larger values

| Species                     | Abbreviation | Assembly | Source   |
|-----------------------------|--------------|----------|----------|
| Homo sapiens                | Hsap         | 36.2     | R        |
| Rattus norvegicus           | Rnor         | RGSC 3.4 | E        |
| Takifugu rubripes           | Trub         | FUGU 4.0 | E        |
| Danio rerio                 | Drer         | Zv6      | E        |
| Drosophila melanogaster     | Dmel         | BDGP 4.3 | R        |
| Anopheles gambia            | Agam         | AgamP3   | R        |
| Apis mellifera              | Amel         | AMEL4.0  | R        |
| Canorhabditis elegans       | Ccle         | WS160    | R        |
| Canorhabditis briggsae      | Cbri         | CB25     | W        |
| Saccharomyces cerevisiae    | Scer         | 2.1      | R        |
| Neurospora crassa OR74A     | Noer         |          | R        |
| Schizosaccharomyces pombe   | Spom         | 1.1      | R        |
| Ustilago maydis 521         | Umay         |          | R        |
| Cryptococcus neoformans v. n. | Cneo      | 1.1      | R        |
| JEC21                       |              |          |          |
| Oryza sativa ssp. japonica  | Osat         | RAP 3    | R        |
| Arabidopsis thaliana        | Atha         | 6.0      | R        |
| Plasmodium falciparum 3D7   | Pfal         | 1.1      | R        |
| Plasmodium berghei str. ANKA | Pber       |          | R        |

R: RefSeq release 20, E: Ensemble release 42, W: WormBase release 160.2
gave identical segmentation results, and lower values resulted in too many scattered blocks). We penalized indels with an infinitely large value to exclude gap columns. Phase-0 introns falling on the boundaries of conserved blocks were excluded. Intron presence and absence in the aligned data was then extracted to produce the data for the likelihood programs.

The final data set comprises 8044 intron sites in 483 sets of orthologs.

**Results** By extending the data of Rogozin et al. (2003), we wanted to examine the effects of increased taxon sampling in the analysis of intron evolution. In the course of that scrutiny, we were also able to refine the findings of previous studies concerning the dynamics of intron evolution.

Figure 4 shows the intron densities for ancestral species. Ancestral nodes such as the bilaterian ancestor (node 4), the ecdysozoan ancestor (node 5), the opisthokont ancestor (node 15) and the bikont ancestor (node 16) all have high intron densities, exceeding previous estimates of Rogozin et al. (2003) and Csőrösi (2005). Honey-bee genes (IHBSC, 2006) indicate a more intron-rich ecdysozoan ancestor (7 introns per gene) than what was known before, and surpass even the generous estimate (6.1 introns per gene) of Roy and Gilbert (2005). (By removing the honey-bee data, our estimate drops by 25%.) Intron density in the bilaterian ancestor is estimated to be almost as high as in humans. Sequences of a handful genes from a certain marine annelid have already indicated that the bilaterian ancestor had at least two-thirds as many introns as vertebrates (Raible et al., 2005).

Most branches are characterized by net intron loss. Figure 5 shows the estimated changes in a few lineages. Intron evolution has been much slower in the vertebrate lineage than in most other lineages: in more than 380 million years since the divergence with fishes, only about 3% of our introns got lost. Fungi, for example, massively trimmed their introns in many lineages as can be seen by the substantial decreases from node 12 onwards, covering a comparable time period. A notable exception is *C. neoformans*, which seems to have gained introns, but that assessment may change if more basidiomycetes are included.

**6 CONCLUSION**

We presented an alignment technique for establishing intron orthology, and a likelihood framework in which intron evolutionary events can be quantified. We described a compression method for the evaluation of the likelihood function, which has been extremely valuable in practice. We also showed that the compression leads to sublinear running times for likelihood evaluation.

We illustrated our methods for analyzing intron evolution with a large and diverse set of eukaryotic organisms. The data set is more comprehensive than any used in other studies published to this day. The data indicate that ancestral eukaryotic genomes were more intron-rich than previous studies suggested.

Many circumstances influence intron loss (Jeffares et al., 2006; Roy and Gilbert, 2006), and realistic likelihood models need to introduce rate variation (Carmel et al., 2005). Usual rate variation models (Felsenstein, 2004) entail multiple evaluations of the likelihood function, and, thus, underline the importance of computational efficiency. We believe that the
proposed methods will help to produce and analyze large data sets even within complicated likelihood models.

ACKNOWLEDGEMENTS

M.Cs. is supported by a grant from the Natural Sciences and Engineering Research Council of Canada. I.B.R. was supported by the NLM/NIH/DHHS Intramural Research Program.

Conflict of Interest: none declared.

REFERENCES

Adachi,J. and Hasegawa,M. (1995) MOLPHY version 2.3: programs for molecular phylogenetics based on maximum likelihood. In: Vol. 28 of Computer Science Monographs. Institute of Statistical Mathematics, Tokyo, Japan, pp. 1–150.

Aldous,D. (2001) Stochastic models and descriptive statistics for phylogenetic trees, from Yule to today. Stat. Sci., 16, 23–34.

Altschul,SR. et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res., 25 (1), 3389–3402.

Berti,T. et al. (2007) WormBase: new content and better access. Nucleic Acids Res., 35 (Suppl. 1), D506–510.

Blum,M. and Francois,O. (2005) On statistical tests of phylogenetic tree evolution. Mol. Biol. Evol., 22, 422–432.

Bieri,T. et al. (2003) Alignment between two multiple alignments. Proceedings of CPM, Vol. 2676 of LNCS. Springer, pp. 254–265.

Marchler-Bauer,A. et al. (2007) CDD: a conserved domain database for interactive domain family analysis. Nucleic Acids Res., 35 (Suppl. 1), D237–240.

McDiarmid,C. (1988) On the method of bounded differences. In: Siemons,J. (ed.) Surveys in Combinatorics. Cambridge University Press, pp. 148–184.

McKenzie,A. and Steel,M. (2000) Distributions of cherries for two models of trees. Math. Biosci., 164, 81–92.

Friedman,N. et al. (2002) A structural EM algorithm for phylogenetic inference. J. Comput. Biol., 9 (2), 331–353.

Guindon,S. and Gascuel,O. (2003) A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. Syst. Biol., 52 (5), 696–704.

Harding,E. (1971) The probabilities of rooted tree-shapes generated by random bifurcation. Adv. Appl. Probab., 3, 44–77.

Heard,SB. (1992) Patterns in tree balance among cladistic, phenetic, and randomly generated phylogenetic trees. Evolution, 46 (6), 1818–1826.

Hubbard,TJ. et al. (2007) Ensemble 2007. Nucleic Acids Res., 35 (Suppl. 1), D610–617.

IBSC (2006) Insights into social insects from the genome of the honey bee Apis mellifera. Nature, 443, 931–949.

Jeffares,DC. et al. (2006) The biology of intron gain and loss. Trends Genet., 22 (1), 16–22.

Keececioglu,J. and Zhang,W. (1998) Aligning alignments. In: Farach-Colton,M. (ed.) Proceedings of CPM, Vol. 1448 of LNCS. Springer, pp. 189–208.

Kosakovský Pond,SL. and Muse,SV. (2004) Column sorting: rapid calculation of the likelihood function. Syst. Biol., 53 (3), 685–692.

Larget,B. and Simon,DL. (1998) Faster likelihood calculations on trees. In: Technical Report 98-02. Department of Mathematics and Computer Science, Duquesne University, Pittsburgh, Pennsylvania, USA.

Ma,B. et al. (2003) Alignment between two multiple alignments. In: Baeza-Yates,R.A., Chávez,E. and Crochemore,M. (eds.) Alignment between two multiple alignments Proceedings of CPM, Vol. 2676 of LNCS. Springer, pp. 254–265.

Marchler-Bauer,A. et al. (2007) CDD: a conserved domain database for interactive domain family analysis. Nucleic Acids Res., 35 (Suppl. 1), D237–240.

McDiarmid,C. (1988) On the method of bounded differences. In: Siemons,J. (ed.) Surveys in Combinatorics. Cambridge University Press, pp. 148–184.

McKenzie,A. and Steel,M. (2000) Distributions of cherries for two models of trees. Math. Biosci., 164, 81–92.

Miller,T. et al. (2002) Estimating amino acid substitution models: a comparison of Dayhoff’s estimator, the resolvent approach and a maximum likelihood method. Mol. Biol. Evol., 19 (1), 8–13.

Nguyen,HD. et al. (2005) New maximum likelihood estimators for eukaryotic intron evolution. PLoS Comput. Biol., 1 (7), e79.

Nielsen,CH. et al. (2004) Patterns of intron gain and loss in fungi. PLoS Biol., 2 (12), e422.

Roy,SW. and Gilbert,W. (2005) Complex early genes. Trends Genet., 21–28.

Roy,SW. and Gilbert,W. (2006) The evolution of spliceosomal introns: patterns, positions and massive, lineage-specific intron loss and gain in eukaryotic evolution. Curr. Biol., 16, 1512–1517.

Roy,SW. et al. (2005) Analysis of evolution of exon-intron structure of eukaryotic genes. Brief. Bioinform., 6 (2), 118–134.

Ribeiro,F. et al. (2005) Vertebrate-type intron-rich genes in the marine annelid Platynereis dumerilii. Science, 310, 1325–1326.

Rogozin,IB. et al. (2003) Remarkable interkingdom conservation of intron positions and massive, lineage-specific intron loss and gain in eukaryotic evolution. Curr. Biol., 13, 1512–1517.

Rogozin,IB. et al. (2005) Analysis of evolution of exon-intron structure of eukaryotic genes. Brief. Bioinform., 6 (2), 118–134.

Rosenberg,NA. (2006) The mean and variance of r-pronged nodes and r-caterpillars in Yule-generated genealogies. Ann. Combinatorics, 10, 129–146.

Roy,SW. and Gilbert,W. (2005) Complex early genes. Proc. Nat. Acad. Sci. USA, 102 (6), 1986–1991.

Roy,SW. and Gilbert,W. (2006) The evolution of splicedosomal introns: patterns, puzzles and progress. Nat. Rev. Genet., 7, 211–221.

Roy,SW. and Penny,D. (2006) Large-scale intron conservation and order-of-magnitude variation in intron loss/gain rates in apicomplexan evolution. Genome Res., 16 (10), 1270–1275.

Roy,SW. and Penny,D. (2007) Patterns of intron loss and gain in plants: Intron loss-dominated evolution and genome-wide comparison of O. sativa and A. thaliana. Mol. Biol. Evol., 24 (1), 171–181.

Stamatakis,AP. et al. (2002) A*ML: A fast program for sequential and parallel phylogenetic tree calculations based on the maximum likelihood method. Proceedings of the IEEE Computer Society Bioinformatics Conference (CSB), 21–28.

Steel,MA. (1994) Recovering a tree from the leaf colours it generates under a Markov model. Appl. Math. Lett., 7, 19–24.

Sverdlov,AV. et al. (2005) Conservation versus parallel gains in intron evolution. Nucleic Acids Res., 33 (6), 1741–1748.

Sverdlov,AV. et al. (2007) A glimpse of a putative pre-intron phase of eukaryotic evolution. Trends Genet., 23 (3), 105–108.

Tatusov,RL. et al. (2003) The COG database: an updated version includes eukaryotes. BMC Bioinformatics, 4, 441.

Valladares,SL. et al. (2005) Splicedosomal introns in the deep-branching eukaryote Trichomonas vaginalis. Proc. Nat. Acad. Sci. USA, 102 (12), 4430–4435.

Zhang,Z. et al. (1999) Post-processing long pairwise alignments. Bioinformatics, 15 (12), 1012–1019.