BIOINFORMATICS

Vol. 19 no. 8 2003, pages 999–1008
DOI: 10.1093/bioinformatics/btg109

Generating consensus sequences from partial order multiple sequence alignment graphs

Christopher Lee[1, 2]

[1]UCLA-DOE Center for Genomics and Proteomics, Molecular Biology Institute and [2]Department of Chemistry and Biochemistry, University of California, Los Angeles, Los Angeles, CA 90095-1570, USA

Received on October 3, 2002; revised on December 10, 2002; accepted on December 20, 2002

ABSTRACT

Motivation: Consensus sequence generation is important in many kinds of sequence analysis ranging from sequence assembly to profile-based iterative search methods. However, how can a consensus be constructed when its inherent assumption—that the aligned sequences form a single linear consensus—is not true?

Results: Partial Order Alignment (POA) enables construction and analysis of multiple sequence alignments as directed acyclic graphs containing complex branching structure. Here we present a dynamic programming algorithm (*heaviest_bundle*) for generating multiple consensus sequences from such complex alignments. The number and relationships of these consensus sequences reveals the degree of structural complexity of the source alignment. This is a powerful and general approach for analyzing and visualizing complex alignment structures, and can be applied to any alignment. We illustrate its value for analyzing expressed sequence alignments to detect alternative splicing, reconstruct full length mRNA isoform sequences from EST fragments, and separate paralog mixtures that can cause incorrect SNP predictions.

Availability: The *heaviest_bundle* source code is available at http://www.bioinformatics.ucla.edu/poa

Contact: leec@mbi.ucla.edu

INTRODUCTION

The general idea of constructing a consensus sequence or statistical profile from a multiple sequence alignment (MSA) is a major theme in many kinds of sequence analysis. It can be considered a basic form of pattern discovery, in which a larger multiple sequence alignment is condensed to a summary of those features that are conserved. It has a wide range of applications ranging from simple to sophisticated. At one extreme, a consensus sequence can be constructed by a 'voting' protocol in which the most frequent letter is selected at each position. Alternatively, a profile allowing several different letters at each position can be constructed, either as a regular expression (e.g. PROSITE Bairoch *et al.*, 1995), a straightforward amino acid frequency profile (e.g. Gribskov *et al.*, 1987; Altschul *et al.*, 1997), or a Hidden Markov Model (Krogh *et al.*, 1994; Eddy, 1996) trained via machine learning methods such as Expectation Maximization and Gibbs Sampling (Lawrence *et al.*, 1993).

Regardless of how the consensus is generated, it rests on a fundamental assumption—that there *exists* a single consensus for the input sequences. Ordinarily, this assumption is justified by the detection of sequence homology within all the sequences in the MSA, for example using Expectation Value statistics as calculated by BLAST (Altschul *et al.*, 1990) and many sequence alignment programs. The presence of non-random similarities throughout the length of a set of sequences can be taken as indicating a common ancestry, in which case the 'consensus' or statistical profile constructed from the MSA should reveal the pattern of selection pressures on the evolution of these sequences from that common ancestor. The profile typically handles two types of sequence change (substitution, and insertion/deletion of single residues). Because such small mutations do not alter the large-scale structure of the sequence, the resulting sequences will align in a simple, linear path that can be represented adequately by a consensus or linear profile string.

Unfortunately, interesting biological sequence comparisons are frequently more complicated (Lecompte *et al.*, 2001). When more complex changes such as duplication, domain translocation, and recombination occur in a set of sequences, the resulting alignment will have complex branching structure that is inconsistent with the very idea that there is any single profile which can represent the alignment. Stated very simply, a pair of multidomain sequences may be homologous in one domain, but unrelated elsewhere, and no single consensus or profile can represent both sequences. Thus, detection of significant similarity is not enough to justify the 'consensus assumption', unless the homology extends over the full length of all sequences.

SYSTEM AND METHODS

Just as we can use graph theory to generalize sequence alignment from working with just linear sequences to work with branched partial orders, we can generalize consensus generation to work with complex alignment structures containing both homology and divergence. We formalize partial order consensus generation as finding an optimal traversal path across the directed graph representing an MSA. For example, existing consensus/profile construction methods can be reformulated as 'Find the path through the alignment with maximum probability' under some specific likelihood model (e.g. 'choose the letter or profile at each position in the alignment which maximizes the likelihood of the observed sequences'). This approach can be applied rigorously even to an MSA graph that contains complex branching. However, in this case *more than one* maximum likelihood traversal will be required to represent the content of the MSA. This raises an algorithmic question: what is the minimum set of linear traversals required to represent the complete PO-MSA?

In this paper, we present a dynamic programming algorithm for consensus sequence generation from partial order alignments, which uses this principle to detect and analyze complex branching structure in multiple sequence alignments. Our algorithm is based on the Partial Order Multiple Sequence Alignment (PO-MSA) data structure for representing a multiple sequence alignment as a graph of nodes (individual sequence letters) and directed edges (their connectivity; Lee *et al.*, 2002), The PO-MSA format contains all the information of a traditional MSA, but represents it as a graph compacted for minimal node and edge counts. A node may have any number of incoming or outgoing edges, each marked with an edge weight reflecting the strength of probabilistic evidence for the existence of that specific edge. Throughout this paper we will refer to a node with no incoming edges as a 'start node', and a node with no outgoing edges as an 'end node'.

ALGORITHM

Consensus generation as graph traversal

The PO-MSA data structure allows us to generalize the notion of 'consensus' construction in the following way. Instead of 'voting' for the most representative letter or profile in each column of a conventional RC-MSA alignment to produce an overall consensus, we reformulate the problem as *choosing the best possible path* through the graph structure of the PO-MSA. Applying this to a simple alignment gives a single traversal that represents the consensus of the aligned sequences (Fig. 1A). On the other hand, if multiple branches are present in the PO-MSA graph G, many paths through the alignment are possible (Fig. 1B). To choose among these paths, we invoke the general principle of maximum likelihood to find the best traversal across the graph, which maximizes the probability of the observations (the aligned sequences). There is a well-established foundation of graph algorithms (e.g. Viterbi algorithm) and probabilistic theory (e.g. Hidden Markov Models) for the solution to this problem.

While these methods can identify an optimal traversal of the graph, it obviously will be representative of only a subset of the aligned sequences (since there are alternative paths). Therefore we extend the consensus generation problem to one of finding the minimum set of traversals needed to represent the structure of the graph. Here we present a simple iterative algorithm that generates multiple consensus traversals by repeatedly constructing a Viterbi (maximum likelihood) traversal, identifying sequences that match this path adequately, adjusting their contributions to the graph edge weights, and iterating this process until no more sequences are left. We refer to the traversal path as a 'consensus', the set of sequences that fit it as a 'bundle', and the traversal algorithm as the 'heaviest bundle' algorithm because it chooses a path by maximum weight.

In addition to generating consensus sequences C_1, $C_2 \ldots$, the top-level function GENERATE_CONSENSI() returns their alignment (in G), and assigns each sequence S_k to a specific consensus (or to none at all). It adds a new sequence C_b to the PO-MSA to represent each consensus, which follows the maximum likelihood path P through G. The pseudocode below shows the overall flow of the algorithm, which we will now describe in detail.

```
GENERATE_CONSENSI($S$, $G$,$\underline{w}$)
{
  b→0
  while (sequences left to be bundled) {
    $P$→ HEAVIEST_BUNDLE($G$, $w$)
    $C_b$→ CREATE_SEQUENCE_ON_PATH($P$, $G$)
    $I$→ ADD_SEQUENCES_TO_BUNDLE($S$,$B$ ,$G$,$C_b$)
    if ($I$ is NULL)
      break
    RESCALE_WEIGHTS($I$, $w$)
    $b$++
  }
  return list of $C_b$
}
```

Dynamic programming traversal algorithm

Consider a PO-MSA graph G consisting of nodes i, j ... and directed edges e_{ij} (to indicate an edge between a pair of nodes (i, j)) with edge weights w_{ij}. Each node is a single letter from one or more aligned sequences. Each edge represents a connection from one letter to the next letter in (at least) one sequence. The weight of a given edge w_{ij} is simply the sum of the weights w_{ijk} of the sequences

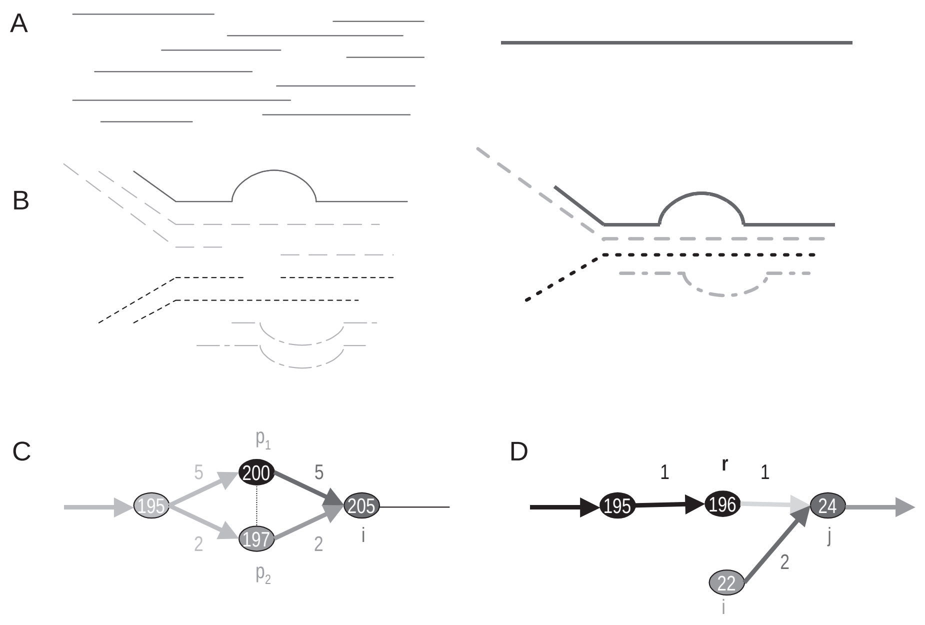

Fig. 1. *Graph traversal construction of consensus sequences.* (A) Graph traversal of a set of sequences that align as a single, unbranched assembly produces a single consensus, representing the maximum likelihood path (bold line, right) through the alignment graph, bypassing 'minority' features such as insertions, deletions and substitutions. (B) Graph traversal of an alignment containing branching structure gives rise to multiple consensus sequences (bold lines, right), representing the divergent paths through the alignment. (C) Dynamic programming local optimality rule for graph traversal. For each node i, the incoming edge with maximum edge weight is selected, and the node is assigned a score equal to the sum of the edge weight and preceding node score (see text). (D) 'Internal termination' occurs when the maximum scoring node in the entire graph (r) has an outgoing edge to another node j, but another node i is selected as the best predecessor for j because its edge has higher weight (see text).

S_k that traverse edge e_{ij}

$$w_{ij} = \sum_{k}^{\forall S_k : i \to j} w_{ijk}.$$

We traverse the graph from left to right; that is, for any two nodes i and j such that there exists a path of directed edges $e_1, e_2, e_3 \dots$ from i to j, it is guaranteed that node i will be visited before node j. Initially all nodes are assigned a score s_i of zero. As each node i is visited, we consider all incoming edges e_{pi}, and select the edge with maximum weight w_{pi} (Fig. 1C). If there is a tie with two (or more) edges having the same weight, the edge e_{pi} is selected that has the predecessor node p with highest score s_p. We then record the optimal edge for node i, and assign the node a score $s_i = s_p + w_{pi}$. We continue to the next node, and repeat until all nodes have been visited. In the very rare case where *both* the edge weights and node scores are tied, one of the edges is selected arbitrarily. It should be noted that if the 'tie' move differs substantially from the chosen path (see *Adjustable Parameters* below for detailed criteria), it will be selected as the best path in the next iteration of the algorithm (see *Iterative Consensus Sequence Generation* below).

```
TRAVERSE_HB($G,w$)
{
  for ($i \to$ all nodes in $G$ from left to right) {
    $p \to$ BEST_PREDECESSOR($w,i$)
    if ($p$ NOT NULL) {
      SAVE_TRACEBACK($i,p$)
      $s_i \to s_p + w_{pi}$
    }
  }
  return $r$ where $s_r = \max_i\{s_i\}$
}
```

Our use of a strictly local rule for selecting the move (giving precedence to maximum w_{pi}, instead of maximum total score $s_p + w_{pi}$) follows from the maximum likelihood principle (i.e. selecting the traversal that maximizes the probability of the observed sequences). For example, consider an alignment in which five sequences follow a directed edge from node i to node j, whereas a single sequence has a large insert *lmno*... p between i and j. Assuming equal weighting $w_{ijk} = 1$ for all six sequences, the correct consensus move is to select node i as the predecessor of j ($w_{ij} = 5$) rather than the long insert *lmno*... p ($w_{pj} = 1$). However, if the length of insert *lmno*... p is greater than five, then $s_p > s_i + 5$ and therefore $s_p + w_{pj} > s_i + w_{ij}$. Thus in this case the maximum total score rule would select the wrong move (it prefers the long insert), while the maximum edge weight rule generates the correct consensus.

To trace back the optimal path, we select the node r with the maximum score ($s_r \geqslant s_i$ for all other nodes i). Beginning at node r, we then trace back the path P of stored best incoming edges until we reach a node l with no incoming edge (i.e. no predecessor nodes). Thus the algorithm combines a completely local rule (for choosing the best incoming edge for each node) with a completely global rule (for choosing the best path endpoint out of all possible paths through G).

Branch completion

We would like our path P to correspond to a complete traversal of the graph, so we require that it begins at a start node and terminates at an end node of G. Since the only possible halting condition for traceback is a node with no incoming edges, it is guaranteed that l is a start node of G. For reasons of symmetry, it is desirable that this also be guaranteed for the right end (node r) of path P. Assuming that all edge weights w_{ij} are defined to be positive, it may appear that r should be an end node. However, a rare condition is possible where this is not true. If r has an outgoing edge e_{rj} to a node j, which also has an incoming edge e_{ij} (from another node i) with a higher edge weight $w_{ij} > w_{rj}$, then i (not r) will be selected as the best predecessor for node j (Fig. 1D). This 'internal termination' condition is easily detected, since it occurs *iff* the maximum scoring node r has outgoing edges e_{rj}.

In this case we apply a *branch completion* procedure as follows. The scores s_i of all nodes $i \neq r$ are set to a negative value to exclude them from consideration as predecessor nodes. For all nodes to the right of r in the graph, we repeat the dynamic programming algorithm with the modification that no node with a negative score can be chosen as the best predecessor of a given node j. Since initially only r has a non-negative score, this guarantees that the new path generated by dynamic programming will begin at r. Since there may be multiple possible paths out of r, generating the best continuation path from r is non-trivial, and requires the full dynamic programming optimization as used above when no 'best starting point' was specified. We use this procedure to find the end node r' with highest score. Since edge weights are required to be positive, it is guaranteed that there exists such a node r' right of r such that $s_{r'} > s_r$. It should be emphasized that it is *not* guaranteed that the node $r*$ with globally maximum score will be an end node (because of the possibility of internal termination, again). Our algorithm avoids this issue by simply selecting the end node r' that has the highest score.

```
HEAVIEST_BUNDLE($G$,$w$)
{
  $r \rightarrow$ TRAVERSE_HB($G$,$w$)
  if ($r$ is not an end node)
    $r \rightarrow$ BRANCH_COMPLETION($r$,$G$,$w$)
    $P \rightarrow$ TRACEBACK($r$,$G$)
    return $P$
}
```

Iterative consensus sequence generation

The maximum likelihood traversal path P identified through G by this procedure is taken to be the principal consensus sequence for G. The top-level function GENERATE_CONSENSI() constructs a new sequence in G that follows path P, and labels it as the consensus sequence C_1. However, C_1 may not be representative of all the sequences in G. We therefore analyze how well the sequences S_k in G fit C_1, and if necessary construct additional consensus sequences for those that do not fit it. We score each sequence S_k's match to C_1 by five adjustable criteria (see **Inclusion Rule** below), and, if above a user-defined threshold, assign its bundle-ID $B_k = C_1$, and rescale its edge weights by $w_{ijk} = w_{ijk} \quad f$ (where f is a user-defined scaling factor, e.g. zero; see **Weight-Rescaling Rule**, below).

```
ADD_SEQUENCES_TO_BUNDLE($S$,$B$,$G$,$C_b$)
{
  $I \rightarrow$ NULL
  for ($k \rightarrow$ all sequences in $S$) {
    if ($B_k$ is NULL && INCLUSION_RULE($S_k$,$C_b$)) {
      $B_k \rightarrow C_b$
      ADD $S_k$ to list $I$
    }
  }
  return $I$
}
```

If there are additional sequences, below the threshold, GENERATE_CONSENSI() simply repeats using the new edge weights, as many times as necessary. This process halts when either: (1) all sequences S_k are assigned a bundle-ID B_k; or (2) *none* of the remaining sequences score above threshold for the most recently generated consensus. In this case, no sequence edge weights would be altered, and consequently repeating the algorithm would result in an infinite loop (producing the same consensus over and over).

Adjustable parameters

The above algorithm is general, and was used without modification to produce all the results in this paper. Its behavior can be controlled by its user-defined parameters, to apply it to very different applications. These parameters fall into three groups (marked in bold, underlined text in the pseudocode):

Edge-weighting rule. Each edge weight w_{ij} is the sum of log-odds weights w_{ijk} from all of the individual sequences S_k that actually traverse that edge, indicating the total strength of evidence for the existence of this edge. The individual edge weight contributions for each sequence can be read from any input, providing a completely general way of controlling the traversal construction process. In this paper we will discuss just two edge weighting schemes: (1) *uniform*: every sequence is initially assigned an equal edge weight contribution; (2) *basecall quality*: the PHRED quality factor (Ewing and Green, 1998) representing the log odds confidence for each basecall in each sequence is used as its edge weight contribution.

Inclusion rule. Five criteria may be specified: *minimum total percent identity* of a sequence S_k versus consensus sequence C_b; *maximum length of internal gaps* in the alignment of sequence S_k versus C_b; *maximum length of terminal branches* (unaligned ends) of sequence S_k versus C_b; *minimum percent identity of the overlap region* of sequence S_k versus C_b; *maximum length of terminal branches of the overlap region* of sequence S_k versus C_b. A sequence must pass all five criteria; however, by choosing a permissive value of a given criterion, any possible intersection of the five criteria may be specified. Other inclusion rules could easily be constructed.

Weight-rescaling rule. Once a sequence is assigned to a consensus bundle, its edge weights w_{ijk} are rescaled by a user-specific parameter f. This controls the degree to which these sequences are separated from subsequent consensus generation steps. For example, if f is set to zero, then these sequences are entirely removed from further consideration.

In this paper we will present two major variations showing how we have adjusted these parameters for different applications:

Paralog separation mode

One application of this algorithm is to separate sets of sequences from an alignment that show evidence of substantial divergence. For example, EST clustering methods (such as UniGene) may mix together ESTs from two or more paralogous genes, and for some applications (such as SNP detection) it is very important to detect and separate this mixture as well as possible. To implement this mode, we combined the following rules (illustrated in Fig. 2A). (1) **Edge-weighting rule**: we use the lod scores of each basecall as edge weights. Using these weights, the heaviest bundling algorithm produces the Viterbi maximum likelihood traversal of the aligned reads. (2) **Inclusion rule**: to assess whether a given sequence S_k is contained in a consensus traversal C_b, we apply a threshold rule consisting of required percent identity (90%), maximum indel (5 nt), and maximum end branch length (20 nt). These parameters are adjustable. (3) **Weight-rescaling rule**: When a sequence S_k is contained in a new consensus C_b, we set its edge weight contributions to zero. Thus, it is completely removed from consideration in the construction of any further consensus traversals.

Isoform generation mode

A second application of this algorithm is to produce the maximum likelihood set of full-length mRNA sequences from the set of transcript fragments for a gene. For example, a UniGene cluster of ESTs and mRNA sequences might contain evidence of alternative splicing/initiation/poly-adenylation. How best to reconstruct the likely full-length transcript isoforms from this collection of fragments? In this paper, we will define the term ‘isoform’ as an end-to-end traversal across the PO-MSA, beginning at a start node and ending at an end node. It should be emphasized that if part(s) of the transcript sequence are missing from the EST data in the PO-MSA, this algorithm cannot reproduce the full-length transcript sequence as defined biologically.

We have implemented a mode for this application as follows (illustrated in Fig. 2B).

Edge-weighting rule. When a cluster contains more than one branch location (e.g. separate alternative splicing events in different parts of the transcript), these separate events may be statistically independent (the frequency of each alternative splice is unaffected by the other alternative splices) or statistically non-independent (e.g. inclusion of one exon by alternative splicing may be accompanied by skipping of another exon elsewhere). The heaviest bundling algorithm, like all dynamic programming computations, assumes statistical independence in the probabilistic model. How then can statistically non-independent branch events be handled correctly? By boosting the edge weight of a single sequence, the algorithm can be forced to follow that sequence as the consensus traversal. Giving a single sequence much higher weight makes the probabilistic model act statistically non-independent, since scoring decisions at many different locations are dominated by the same highly-weighted sequence. We multiply the edge weights of this template sequence by a factor f chosen to be much larger than the total number of sequences in an alignment (for this paper, 1000 was used). To choose a template sequence, we simply select the longest sequence that has *not* yet been assigned to a consensus traversal. In this way, if a full-length mRNA sequence is available, it is selected. Otherwise, the longest EST showing the longest observed traversal is selected.

At the same time, we wish to generate a properly weighted consensus sequence even in this region (rather

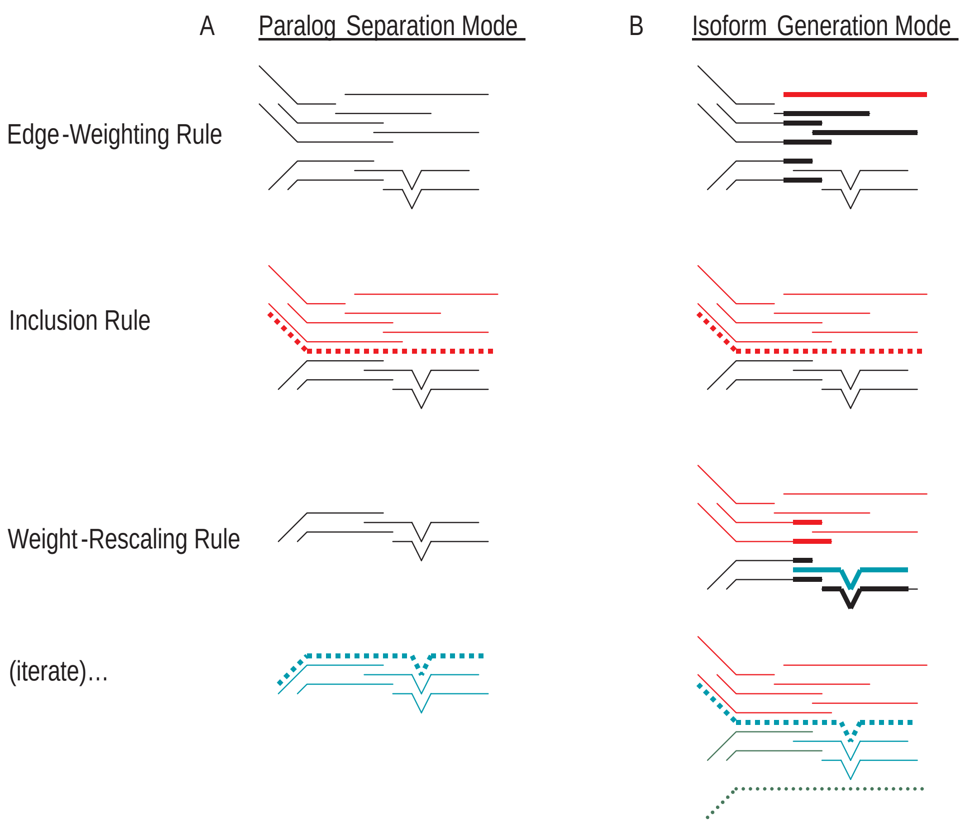

Fig. 2. Paralog separation vs isoform generation modes. Comparison of the Traversal, Inclusion, and Separation Rules for (A) Paralog Separation Mode versus (B) Isoform Generation Mode (the template sequence and aligned regions whose edge weights are up-weighted for Isoform Generation are indicated by bold lines). The consensus traversals produced are shown as dotted lines; sequences assigned to each consensus are colored the same as the consensus. See text for details.

than automatically including sequencing errors which may be present in the template sequence). To do so, we select a subset of sequences that follow very closely the same traversal as the template sequence, by applying a set of criteria (93% identity within the region of alignment to the template; maximum of 5 nt indel as measured by 'overlap alignment' scoring). Within the region where these sequences align to the template, their edge weights are multiplied by f. Elsewhere, their weighting is not increased, and all other sequences' weighting are unchanged. In this way, the heaviest bundle algorithm is forced to follow the template sequence, but a properly weighted consensus sequence is generated both within the region where it aligns (from it and the sequences that follow it closely, all given an equivalent, high weight), and elsewhere in the alignment (regular weighting for all sequences).

Inclusion rule. We use the same inclusion rule as in the Paralog Separation Mode, to select the sequences that will be assigned as belonging to a particular consensus.

Weight-rescaling rule. Since our edge-weighting rule above incorporates statistical non-independence whenever there is direct evidence for it (i.e. a sequence that specifically couples two or more distinct alternative splicing events), in the absence of such evidence our default likelihood model should assume statistical independence of separate branching events. For example, in the absence of any evidence of coupling for a series of alternative splicing events, isoforms should be generated by combining one alternative splice 'minor form' (i.e. the less probable form at that site) with the 'major form' splices from the rest of the transcript. This produces the maximum likelihood isoforms that can explain the observations. This can be accomplished by a trivial modification of the weight-rescaling rule. Rather than setting sequences' weights to zero after they are assigned to a bundle, we simply reset them to their original weighting (removing the temporarily high weighting they were assigned in the Edge-weighting Rule). Thus, they will continue to contribute, with the proper probabilistic weighting, to the determination of the ‘major form’ in different regions of the alignment. We then re-call the Edge-weighting Rule above with a new template sequence chosen from the sequences that have not yet been assigned to any bundle.

In this paper we limited the reporting of consensus isoform sequences to those supported by at least one experimental sequence (in addition to the template sequence) in the edge-weighting step.

Computational complexity. Since the dynamic programming algorithm only needs to visit each node in G once, in a predefined order, its time complexity for both the forward traversal and traceback is simply $O(N)$, where N is the number of nodes in G. Taking into account the number of incoming edge comparisons that have to be performed per node, the total time complexity is $O(n_{\text{ave}}N)$, where n_{ave} is a variable denoting the average number of incoming edges per node in a given alignment, typically one to four for the applications described in this paper. Since only the optimal predecessor and score need to be stored at each node, the memory complexity is also $O(N)$. In our implementation, the nodes are automatically stored in left-to-right order by virtue of how the PO-MSA is constructed. Thus there is no need to pre-sort the nodes to put them in order.

Branch completion increases the time complexity of this algorithm, but not substantially. First of all, since dynamic programming only needs to be repeated for the region to the *right* of the internal termination point, this adds only a fraction of the original computation time. Moreover since the branch completion algorithm guarantees a right-terminal node, this extra search need be performed only once. Thus the total computational complexity remains $O(N)$.

IMPLEMENTATION

This algorithm has been implemented as a C program utilizing the POA library previously described (Lee *et al.*, 2002). It is included in the existing POA program as a command-line option. The source code compiles and runs on a wide variety of UNIX and Windows platforms, and can be downloaded from http://www.bioinformatics.ucla.edu/poa.

Biological sequence analysis applications

EST consensus assembly. To test the algorithm’s consensus generation on a standard case, we used a set of 40 EST sequences from UniGene cluster Hs.7086 (Schuler, 1997) in which only a single consensus was found (indicating absence of branching in the alignment), and compared with the consensus sequence produced by the well-known programs PHRED and PHRAP (Ewing *et al.*, 1998; Ewing and Green, 2000). There was only a single nucleotide difference between the

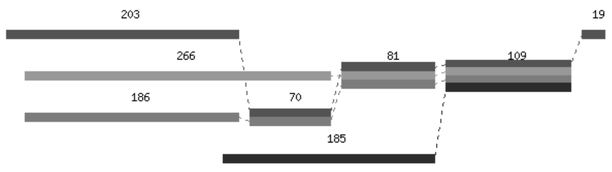

Fig. 3. Heaviest bundle analysis of UniGene EST cluster Hs.258817. EST sequences from UniGene cluster Hs.258817 were aligned by POA, analyzed by the heaviest bundle algorithm (in Paralog Separation Mode), and submitted as a PIR formatted file to the online POAVIZ visualizer tool (http://www.bioinformatics.ucla.edu/poa), showing the four heaviest bundle consensus sequences only. The length of each block (in nucleotides) is indicated above it.

consensus sequences produced by heaviest bundling and PHRED/PHRAP (99.9% identity). This difference was located in the 5′ region of the alignment where only a single EST (Hs#S416368) contributed to the consensus, and resulted from the fact that its sequence in UniGene has this one difference versus the basecall generated by PHRED from the chromatogram trace file for this EST. In other words, this difference arose in the original basecalling of the sequence in UniGene, not from the heaviest bundling algorithm. In the rest of the alignment, where there were at least two EST sequences aligned in each column, the heaviest bundle consensus was identical to the PHRED/PHRAP consensus.

Alignment complexity analysis

This algorithm provides a useful analytic tool for evaluating the structural complexity of any multiple sequence alignment from POA, CLUSTAL (Higgins and Sharp, 1988), or other alignment methods. The number of consensus traversals produced is a basic metric of the structural complexity of the alignment.

Figure 3 illustrates the heaviest bundle analysis of UniGene cluster Hs.258817 (16 ESTs), which reveals that this cluster of ESTs contains four distinct branches diverging at their 5’ ends. All of them match over large regions (ranging in size from 109 to 260 nt), but have even larger regions of divergence (ranging from 185 to 266 nt). The heaviest bundle analysis reveals this structure clearly and concisely; by contrast, looking at the alignment to discern this structure is time-consuming and requires care (even for this relatively small EST cluster, the PIR alignment is 23 000 characters long).

Indeed, heaviest bundle analysis enables completely automated visualization of the structure of any alignment file (e.g. CLUSTAL), using the POAVIZ visualization tool (Grasso *et al.*, 2003 available at http://www.bioinformatics.ucla.edu/poa).

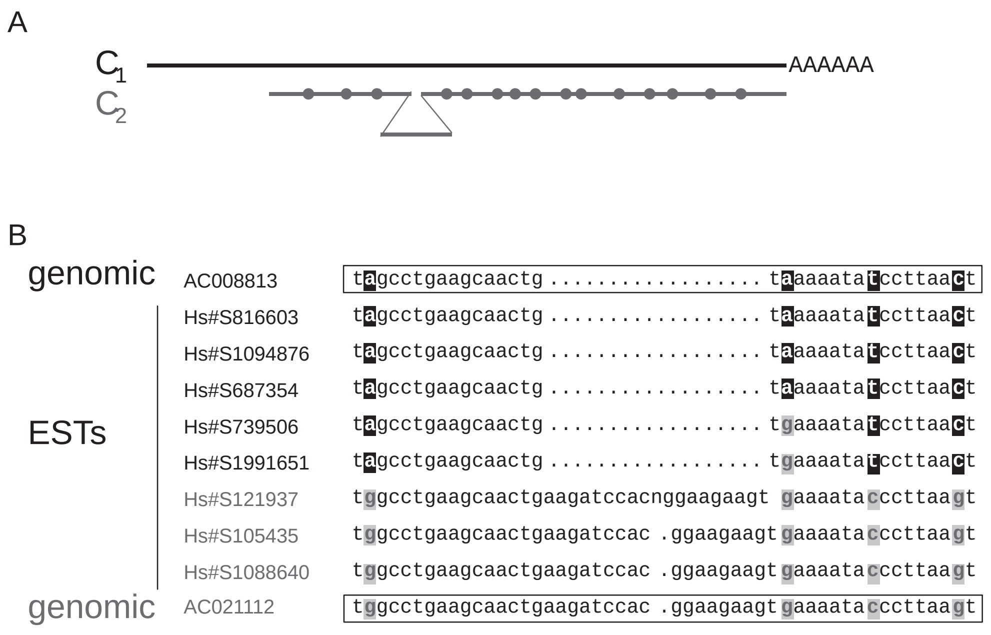
A
C1
C2
AAAAAA
B
genomic
AC008813
tagcctgaagcaactg .... taaaaatatccttaact
Hs#S816603
tagcctgaagcaactg .... taaaaatatccttaact
Hs#S1094876
tagcctgaagcaactg .... taaaaatatccttaact
Hs#S687354
tagcctgaagcaactg .... taaaaatatccttaact
ESTs
Hs#S739506
tagcctgaagcaactg .... tgaaaatatccttaact
Hs#S1991651
tagcctgaagcaactg .... tgaaaatatccttaact
Hs#S121937
tggcctgaagcaactgaagatccacnggaagaagt gaaaataccccttaagt
Hs#S105435
tggcctgaagcaactgaagatccac .ggaagaagt gaaaataccccttaagt
Hs#S1088640
tggcctgaagcaactgaagatccac .ggaagaagt gaaaataccccttaagt
genomic
AC021112
tggcctgaagcaactgaagatccac .ggaagaagt gaaaataccccttaagt

Fig. 4. Paralog separation of UniGene EST cluster Hs.104607. (A) Paralog Separation Mode produces two consensus sequences C_1 and C_2. (B) Comparison of detailed evidence from part of the EST alignment. Sequences assigned to each consensus are colored the same as the consensus (C_1 black; C_2 red). Mapping of these consensus sequences onto the human genomic sequences reveals that they match two different genomic contig locations (C_1 to AC008813; C_2 to AC021112). SNP detection from these sequences (which are merged in UniGene cluster Hs.104607) would have produced 15 spurious SNPs (four of which are highlighted by boxes here).

Paralog separation mode

Figure 4 shows an example of a UniGene cluster (Hs. 104607) that was separated into two distinct bundles by the Paralog Separation Mode. For SNP candidate discovery, we only compared sequences that were within a single bundle, to reduce the probability of paralog contamination. Subsequent mapping of these ESTs onto the human genome sequence showed that the two groups in fact mapped to two distinct genomic locations. Use of the heaviest bundle algorithm thus avoided 15 spurious SNPs (four of which are shown in Fig. 4) that would have been reported with strong scores based on the UniGene cluster alignment. After paralog separation, SNP evidence calculation (Irizarry *et al.*, 2000) identified nine polymorphisms with lod score greater than three. Thus the SNP error rate due to paralog contamination in this cluster would have been greater than 60% without application of the heaviest bundling algorithm.

Isoform generation mode

The heaviest bundle algorithm can also be applied to a very different biological problem: reconstructing the set of full-length isoform sequences based on EST fragments from a mix of alternatively spliced transcripts of a gene. UniGene cluster Hs.1162 illustrates this process (Fig. 5). Two alternative splices A/a and B/b (where the capital letters refer to the major form at each position) are observed in this cluster (Modrek *et al.*, 2001). There is a full length mRNA representing the form *AB*, and EST fragments displaying splices A, B, a and b separately (plus other, constitutive splices in the gene). Our Isoform Generation Mode first constructed a full-length form *AB* (based on both the mRNA and ESTs), then *aB*, and then *Ab*. If no other isoforms were needed to explain the observed data, the algorithm would halt at this point. In Hs.1162 additional ESTs are observed containing *both* splice *a* and splice *b*, so the algorithm also produced the additional full-length isoform *ab*. The four isoforms generated had slight differences in the length of poly-adenylation at their 3′ ends, due to variation in the length of poly-A in the ESTs representing each of the four isoforms. To test the algorithm's ability to reconstruct full-length isoforms entirely from fragment data, we removed all mRNA sequences from the alignment, and repeated Isoform Generation from ESTs only. The algorithm produced the same four isoforms (*AB*, *aB*, *Ab*, *ab*), which matched the isoforms based on the mRNAs (Fig. 5A). We will present elsewhere a detailed statistical analysis of the accuracy of alternative splice isoform construction from EST fragment data.

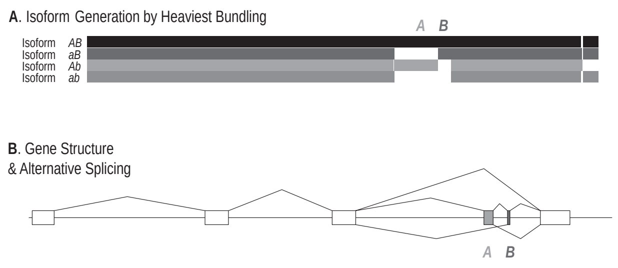
A. Isoform Generation by Heaviest Bundling
A B
Isoform AB
Isoform aB
Isoform Ab
Isoform ab
B. Gene Structure & Alternative Splicing
A B

Fig. 5. Isoform generation for UniGene EST cluster Hs.1162. Comparison of (A) Isoform Generation by the heaviest bundle algorithm, *versus* (B) the full genomic structure of exons, introns, and alternative splices for this gene (HLA-DMB) (Shaman *et al.*, 1995; Modrek *et al.*, 2001). The algorithm correctly constructs four isoforms that include or exclude combinations of two regions *A* and *B*. These correspond to two exons have previously been shown to be alternatively spliced in the expressed sequence data. The Isoform Generation Mode produces the same four isoforms even when mRNA sequences are removed, forcing the algorithm to reconstruct the isoforms entirely from EST fragment data (see text).

DISCUSSION

The biological sequence applications considered briefly above (EST paralog separation and consensus generation) have been analyzed and studied very extensively by previous researchers. The work described in this paper is a minor extension of that large body of research. Previous research has elucidated the construction of EST clusters (Schuler, 1997; Christoffels *et al.*, 2001), alignments, gene indexes (Liang *et al.*, 2000a,b; Quackenbush *et al.*, 2001), consensus sequences (Quackenbush *et al.*, 2000), and identification of alternative isoforms (Burke *et al.*, 1998; Harrison *et al.*, 2002; Kan *et al.*, 2002). Much work has also been devoted to the mapping of expressed sequence data (Zhuo *et al.*, 2001), analysis and visualization (Haas *et al.*, 2000; Muilu *et al.*, 2001) of the interrelationships among these sequences.

Discerning and exploiting the graph structure of biological data is a growing theme in bioinformatics, and seems likely to become increasingly important. Graph representation and algorithms are used throughout Hidden Markov Model methods (for a general review see Eddy, 1996). Graph analysis and traversal algorithms have been applied to pathway construction (Mittenthal, 1997), genome assembly (Myers *et al.*, 2000; Pevzner *et al.*, 2001; Batzoglou *et al.*, 2002), alternative splicing (Heber *et al.*, 2002), phylogeny inference (Strimmer and Moulton, 2000), protein interaction networks (Tong *et al.*, 2002), microarray gene expression analysis (Hartemink *et al.*, 2001; Zhou *et al.*, 2002), peptide identification from mass spectrometry (Dancik *et al.*, 1999), and many other applications. Graph properties such as partial order have been proposed as an essential consistency criterion for evaluating multiple sequence alignments (Morgenstern *et al.*, 1996). Partial Order Alignment (POA; Lee *et al.*, 2002) incorporated partial order graph structure directly into sequence alignment representation and dynamic programming optimal alignment algorithms.

We have used POA and the heaviest bundling algorithm extensively in our previously reported SNP detection (Irizarry *et al.*, 2000; Hu *et al.*, 2002) of over 90,000 human EST clusters, and alternative splicing analysis of the human transcriptome (Modrek *et al.*, 2001; Xu *et al.*, 2002).

The heaviest bundle algorithm has a number of disadvantages. First, it is a heuristic, non-optimal algorithm. It does not provide a definitive guarantee that a particular cost function will be optimized. A second failing is that the algorithm's behavior depends on parameters such as %identity, gap size etc. which are used as thresholds for bundling. It would be better to construct a full probabilistic model of the underlying process emitting the observed sequences, and use properly calculated statistical confidence measures instead of these threshold parameters.

ACKNOWLEDGEMENTS

I wish to thank S. Galbraith, C. Grasso, G. Hu, D.S. Parker and Y. Xing for their helpful discussions and comments on this work. This work was supported by the National Institute of Mental Health and the National Institute of Neurological Disorders and Stroke (Grant #: MH65166), National Science Foundation grant IIS-0082964, and Department of Energy grant DEFG0387ER60615.

SUPPLEMENTARY DATA

For Supplementary data, please refer to *Bioinformatics* online.

REFERENCES

Altschul,S.F., Gish,W., Miller,W., Myers,E.W. and Lipman,D.J. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Altschul,S.F., Madden,T.L., Schäffer,A.A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Bairoch,A., Bucher,P. and Hofmann,K. (1995) The PROSITE datatase, its status in 1995. *Nucleic Acids Res.*, **24**, 189–196.

Batzoglou,S., Jaffe,D.B., Stanley,K., Butler,J., Gnerre,S., Mauceli,E., Berger,B., Mesirov,J.P. and Lander,E.S. (2002) ARACHNE: a whole-genome shotgun assembler. *Genome Res.*, **12**, 177–189.

Burke,J., Wang,H., Hide,W. and Davison,D.B. (1998) Alternative gene form discovery and candidate gene selection from gene indexing projects. *Genome Res.*, **8**, 276–290.

Christoffels,A., van Gelder,A., Greyling,G., Miller,R., Hide,T. and Hide,W. (2001) STACK: sequence tag alignment and consensus knowledgebase. *Nucleic Acids Res.*, **29**, 234–238.

Dancik,V., Addona,T.A., Clauser,K.R., Vath,J.E. and Pevzner,P.A. (1999) De novo peptide sequencing via tandem mass spectrometry. *J. Comp. Biol.*, **6**, 327–342.

Eddy,S.R. (1996) Hidden Markov models. *Curr. Opin. Struct. Biol.*, **6**, 361–365.

Ewing,B. and Green,P. (1998) Base-calling of automated sequencer traces using phred. II. Error probabilities. *Genome Res.*, **8**, 186–194.

Ewing,B. and Green,P. (2000) Analysis of expressed sequence tags indicates 35 000 human genes. *Nat. Genet.*, **25**, 232–234.

Ewing,B., Hillier,L., Wendl,M.C. and Green,P. (1998) Base-calling of automated sequencer traces using phred. I. Accuracy assessment. *Genome Res.*, **8**, 175–185.

Grasso,C., Quist,M., Ke,K. and Lee,C. (2003) POAVIZ: a partial order multiple sequence alignment visualizer. *Bioinformatics*, in press.

Gribskov,M., McLachlan,A.D. and Eisenberg,D. (1987) Profile analysis: detection of distantly related proteins. *Proc. Natl Acad. Sci. USA*, **84**, 4355–4358.

Haas,S.A., Beissbarth,T., Rivals,E., Krause,A. and Vingron,M. (2000) GeneNest: automated generation and visualization of gene indices. *Trends Genet.*, **16**, 521–523.

Harrison,P.M., Kumar,A., Lang,N., Snyder,M. and Gerstein,M. (2002) A question of size: the eukaryotic proteome and the problems in defining it. *Nucleic Acids Res.*, **30**, 1083–1090.

Hartemink,A.J., Gifford,D.K., Jaakkola,T.S. and Young,R.A. (2001) Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. *Pac. Symp. Biocomput.*, **6**, 422–433.

Heber,S., Alekseyev,M., Sze,S.H., Tang,H. and Pevzner,P.A. (2002) Splicing graphs and EST assembly problem. *Bioinformatics*, **18**, S181–S188.

Higgins,D.G. and Sharp,P.M. (1988) CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene*, **73**, 237–244.

Hu,G., Modrek,B., Stensland,H.M.F.R., Saarela,J., Pajukanta,P., Kustanovich,V., Peltonen,L., Nelson,S.F. and Lee,C. (2002) Efficient discovery of single-nucleotide polymorphisms in coding regions of human genes. *Pharmacogenomics J.*, **2**, 236–242.

Irizarry,K., Kustanovich,V., Li,C., Brown,N., Nelson,S., Wong,W. and Lee,C. (2000) Genome-wide analysis of single-nucleotide polymorphisms in human expressed sequences. *Nat. Genet.*, **26**, 233–236.

Kan,Z., States,D. and Gish,W. (2002) Selecting for functional alternative splices in ESTs. *Genome Res.*, **12**, 1837–1845.

Krogh,A., Brown,M., Mian,I.S., Sjolander,K. and Haussler,D. (1994) Hidden Markov models in computational biology: applications to protein modeling. *J. Mol. Biol.*, **235**, 1501–1531.

Lawrence,C.E., Altschul,S.F., Boguski,M.S., Liu,J.S., Neuwald,A.F. and Wootton,J.C. (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, **262**, 208–214.

Lecompte,O., Thompson,J.D., Plewniak,F., Thierry,J. and Poch,O. (2001) Multiple alignment of complete sequences (MACS) in the post-genomic era. *Gene*, **270**, 17–30.

Lee,C., Grasso,C. and Sharlow,M. (2002) Multiple sequence alignment using partial order graphs. *Bioinformatics*, **18**, 452–464.

Liang,F., Holt,I., Pertea,G., Karamycheva,S., Salzberg,S.L. and Quackenbush,J. (2000a) Gene Index analysis of the human genome estimates approximately 120 000 genes. *Nat. Genet.*, **25**, 239–240.

Liang,F., Holt,I., Pertea,G., Karamycheva,S., Salzberg,S.L. and Quackenbush,J. (2000b) An optimized protocol for analysis of EST sequences. *Nucleic Acids Res.*, **28**, 3657–3665.

Mittenthal,J.E. (1997) An algorithm to assemble pathways from processes. *Pac. Symp. Biocomput.*, **2**, 292–303.

Modrek,B., Resch,A., Grasso,C. and Lee,C. (2001) Genome-wide analysis of alternative splicing using human expressed sequence data. *Nucleic Acids Res.*, **29**, 2850–2859.

Morgenstern,B., Dress,A. and Werner,T. (1996) Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc. Natl Acad. Sci. USA*, **93**, 12 098–12 103.

Muilu,J., Rodriguez-Tome,P. and Robinson,A. (2001) GBuilder—an application for the visualization and integration of EST cluster data. *Genome Res.*, **11**, 179–184.

Myers,E.W., Sutton,G.G., Delcher,A.L., Dew,I.M., Fasulo,D.P., Flanigan,M.J., Kravitz,S.A., Mobarry,C.M., Reinert,K.H.J., Remington,K.A. *et al.* (2000) A whole-genome assembly of *Drosophila*. *Science*, **287**, 2196–2204.

Pevzner,P.A., Tang,H. and Waterman,M.S. (2001) An Eulerian path approach to DNA fragment assembly. *Proc. Natl Acad. Sci. USA*, **98**, 9748–9753.

Quackenbush,J., Liang,F., Holt,I., Pertea,G. and Upton,J. (2000) The TIGR Gene Indices: reconstruction and representation of expressed gene sequences. *Nucleic Acids Res.*, **28**, 141–145.

Quackenbush,J., Cho,J., Lee,D., Liang,F., Holt,I., Karamycheva,S., Parvizi,B., Pertea,G., Sultana,R. and White,J. (2001) The TIGR Gene Indices: analysis of gene transcript sequences in highly sampled eukaryotic species. *Nucleic Acids Res.*, **29**, 159–164.

Schuler,G. (1997) Pieces of the puzzle: expressed sequence tags and the catalog of human genes. *J. Mol. Med.*, **75**, 694–698.

Shaman,J., von Scheven,E., Morris,P., Chang,M.Y. and Mellins,E. (1995) Analysis of HLA-DMB mutants and -DMB genomic structure. *Immunogenetics*, **41**, 117–124.

Strimmer,K. and Moulton,V. (2000) Likelihood analysis of phylogenetic networks using directed graphical models. *Mol. Biol. Evol.*, **17**, 875–881.

Tong,A.H.Y., Drees,B., Nardelli,G., Bader,G.D., Brannetti,B., Castagnoli,L., Evangelista,M., Ferracuti,S., Nelson,B., Paoluzi,S. *et al.* (2002) A combined experimental and computational strategy to define protein interaction networks for peptide recognition modules. *Science*, **295**, 321–324.

Xu,Q., Modrek,B. and Lee,C. (2002) Genome-wide detection of tissue-specific alternative splicing in the human transcriptome. *Nucleic Acids Res.*, **30**, 3754–3766.

Zhou,X., Kao,M.-C.J. and Wong,W.H. (2002) Transitive functional annotation by shortest path analysis of gene expression data. *Proc. Natl Acad. Sci. USA*, in press.

Zhuo,D., Zhao,W.D., Wright,F.A., Yang,H.Y., Wang,J.P., Sears,R., Baer,T., Kwon,D.H., Gordon,D., Gibbs,S. *et al.* (2001) Assembly, annotation, and integration of UNIGENE clusters into the human genome draft. *Genome Res.*, **11**, 904–918.