Sequence analysis

GMAP: a genomic mapping and alignment program for mRNA and EST sequences

Thomas D. Wu[1],* and Colin K. Watanabe[2]

[1]Department of Bioinformatics and [2]Department of Corporate Information Technology, Genentech, Inc., South San Francisco, CA 94080, USA

ABSTRACT

Motivation: We introduce GMAP, a standalone program for mapping and aligning cDNA sequences to a genome. The program maps and aligns a single sequence with minimal startup time and memory requirements, and provides fast batch processing of large sequence sets. The program generates accurate gene structures, even in the presence of substantial polymorphisms and sequence errors, without using probabilistic splice site models. Methodology underlying the program includes a minimal sampling strategy for genomic mapping, oligomer chaining for approximate alignment, sandwich DP for splice site detection, and microexon identification with statistical significance testing.

Results: On a set of human messenger RNAs with random mutations at a 1 and 3% rate, GMAP identified all splice sites accurately in over 99.3% of the sequences, which was one-tenth the error rate of existing programs. On a large set of human expressed sequence tags, GMAP provided higher-quality alignments more often than BLAT did. On a set of *Arabidopsis* cDNAs, GMAP performed comparably with GeneSeqer. In these experiments, GMAP demonstrated a several-fold increase in speed over existing programs.

Availability: Source code for GMAP and associated programs is available at http://www.gene.com/share/gmap

Contact: twu@gene.com

Supplementary information: http://www.gene.com/share/gmap

INTRODUCTION

Mapping and alignment of cDNA sequences—both messenger RNAs (mRNAs) and expressed sequence tags (ESTs)—onto the genome has become a central procedure in genome research. The resulting cDNA–genomic alignments not only reveal the intron–exon structure of genes, but also facilitate the study of splicing mechanics and such transcript-based phenomena as alternative splicing, single nucleotide polymorphisms, and cDNA insertions and deletions (Jiang and Jacob, 1998; Irizarry *et al.*, 2000; Kan *et al.*, 2001, 2002; Zavolan *et al.*, 2002; Modrek and Lee, 2002; Clamp *et al.*, 2003; Wheeler *et al.*, 2003; Drabenstot *et al.*, 2003; Kim *et al.*, 2004; Florea *et al.*, 2005).

To address these needs, programs such as SSAHA (Ning *et al.*, 2001) have been introduced to map cDNA sequences to a genome. Other programs have been developed to align a cDNA to a given genomic segment, including EST_GENOME (Mott, 1997),

dds/gap2 (Huang, 1996), SIM4 (Florea *et al.*, 1998), Spidey (Wheelan *et al.*, 2001), GeneSeqer (Usuka *et al.*, 2000; Schlueter *et al.*, 2003) and MGAlign (Lee *et al.*, 2003; Ranganathan *et al.*, 2003). Finally, some recent integrated programs, such as BLAT (Kent, 2002) and SQUALL (Ogasawara and Morishita, 2002), perform both genomic mapping and alignment.

Despite the availability of these programs, achieving perfection in cDNA–genomic alignment has been surprisingly elusive. Studies of existing programs have revealed various types of errors in identifying gene structures and splice sites (Haas *et al.*, 2002). In compiling a database of EST-based splice sites, researchers have reportedly had to resort to manual curation of alignments to obtain the correct results (Burset *et al.*, 2001). Difficulties generally arise when a cDNA sequence differs from its corresponding genomic exons, due to polymorphisms, mutations or sequencing errors. Sequencing errors are especially prevalent in ESTs, where error rates are estimated to be 1.5% for high-quality sequences (Zhuo *et al.*, 2003) and 3–4% overall (Richterich, 1998). Such sequencing errors, especially near exon–exon junctions, can complicate the detection of splice sites.

One approach to this situation has been to combine information across various alignments (Birney *et al.*, 2004; Haas *et al.*, 2003; Brendel *et al.*, 2004) or even multiple sources of evidence (Allen *et al.*, 2004) to arrive at a consensus answer. However, since such programs depend ultimately upon the original solutions generated by cDNA–genomic alignment programs, advances in the underlying alignment methodology are still important.

In this paper, we introduce an integrated genomic mapping and alignment program called GMAP (Genomic Mapping and Alignment Program). In contrast to programs designed primarily to run in client/server mode, such as BLAT and SQUALL, our program operates as a traditional standalone program. GMAP provides not only improved performance over existing programs in terms of speed and accuracy, but also enhanced functionality. The functionality provided by GMAP allows a user to: (1) map and align a single cDNA interactively against a large genome in about a second, without the startup time of several minutes typically needed by existing mapping programs; (2) switch arbitrarily among different genomes, without the need for a pre-loaded server dedicated to each genome; (3) run the program on computers with as little as 128 MB of RAM (random access memory); (4) perform high-throughput batch processing of cDNAs by using memory mapping and multithreading when appropriate memory and hardware are available; (5) generate accurate gene models, even in the presence of substantial polymorphisms and sequence errors; (6) locate splice sites accurately without the use

---

*To whom correspondence should be addressed.

of probabilistic splice site models, allowing generalized use of the program across species; (7) detect statistically significant microexons and incorporate them into the alignment; and (8) handle mapping and alignment tasks on genomes having alternate assemblies, linkage groups or strains.

In the remainder of the paper, we review existing work on cDNA–genomic mapping and alignment, and describe the methods underlying GMAP. Next we provide examples of how these methods in GMAP lead to improved splice site and gene structure prediction. Then we compare the performance of GMAP with existing programs in three large-scale experiments. In experiment 1, we test for robustness to sequence error by using test sets of human mRNAs with computationally simulated sequence errors. In experiment 2, we examine mapping and alignment quality for human ESTs with naturally occurring sequence errors. In experiment 3, we evaluate the performance of GMAP on another species, namely, the plant *Arabidopsis thaliana*. Finally, we describe the implementation of GMAP and additional features provided by the program.

RELATED WORK

One approach to cDNA–genomic alignment has been to use general sequence alignment programs, such as BLAST (Altschul *et al.*, 1990), and then to assemble the resulting hits into gene structures (Gelfand *et al.*, 1996; Wiehe *et al.*, 2001; Milanesi and Rogozin, 2003; Zhang, 2003; Yeo *et al.*, 2004). However, the cDNA–genomic alignment problem is important enough to warrant programs specialized for the task. The particular problem that arises in cDNA–genomic alignment is the presence of introns, which appear as large genomic gaps of up to hundreds of thousands of nucleotides in length. Introns have characteristic patterns at their splice sites, which cDNA–genomic alignment programs must take into account. About 99% of introns are bounded on their ends by the canonical dinucleotide pair GT–AG; the remainder have a semi-canonical dinucleotide pair GC–AG or AT–AC, or another, non-canonical dinucleotide pair (Burset *et al.*, 2000). Probabilistic patterns of conservation are also seen at positions further away from the intron–exon boundary (Mount, 1981; Senapathy *et al.*, 1990; Solovyev, 2002).

Existing programs for cDNA–genomic mapping and alignment, cited in the Introduction, provide a foundation for further advances. In particular, GMAP draws upon three fundamental concepts introduced by earlier programs. First, GMAP uses an oligomer index table for genomic mapping. Second, GMAP takes a hierarchical approach to genomic alignment, by first computing an approximate alignment and then filling in the details. Finally, like almost all existing alignment programs, GMAP applies specific methods tailored for detecting splice sites and for incorporating them into the alignment.

Although essentially all cDNA–genomic mapping and alignment programs share these fundamental building blocks, they differ in their particular methods for implementing them; it is these methodological choices that largely account for differences in their performance. In the Algorithm section, we provide a detailed description of the specific methods underlying GMAP; in the rest of this section, we summarize the basic similarities and differences of our methods relative to existing ones.

Genomic mapping

Genomic mapping can be accomplished rapidly because of the near-identity between a cDNA sequence and its corresponding genomic exons, which manifests as regions of exact matches. Existing programs exploit this fact either by finding clusters of relatively short oligomers, such as 11-mers (BLAT) or 14-mers (SSAHA and SQUALL), or by using fewer long oligomers. The long oligomer approach is exemplified by MGAlign (Ranganathan *et al.*, 2003): although it does not perform mapping on a genomic scale, it initially aligns a cDNA to a given genomic segment by scanning 20-mers from the ends of the cDNA. Similarly, rapid mapping is provided by MUMmer (Delcher *et al.*, 1999, 2002), which uses suffix trees (Manber and Myers, 1993) to find long unique matches between genomes, and MegaBlast (Zhang *et al.*, 2000), which uses 28-mers to identify sequence matches.

Existing cDNA–genomic mapping programs that use an oligomer index on a genomic scale begin by pre-loading the index into memory, which means that these programs not only have a long startup time, but also require computers with large amounts of dedicated RAM. For example, SQUALL requires 12 GB of RAM, and the standalone version of BLAT requires 8 GB of RAM in order to map a cDNA sequence onto the entire human genome. The startup time for the standalone version of BLAT is several minutes, which makes it inconvenient for a researcher who wishes to map a single cDNA sequence to a genome, or who wishes to switch quickly among different genomes or versions of a genome. Therefore, BLAT typically runs in a client–server mode, in which a dedicated server for a particular genome keeps its genomic oligomer files resident in RAM. A BLAT server, which also requires several minutes of startup time, needs 1.2 GB of RAM to process the human genome, and must be kept running continuously to answer queries from a client computer.

In contrast, GMAP is a standalone program that has been designed to handle individual queries rapidly, with essentially no startup time. Instead of pre-loading the entire oligomer index file into memory, GMAP looks up oligomers as needed directly from the file. Because access to files is much slower than to memory, our file-based strategy is enabled by a minimal sampling strategy that attempts to perform as few oligomer lookups as possible, while still mapping reliably to an entire genome. Our sampling strategy involves more than scanning long oligomers from the ends of a cDNA to find a matching pair. Because our mapping universe is an entire genome, we must safeguard against false mapping results from the initial matching pair, which can arise due to paralogs, pseudogenes and segmental duplications in the genome (Wheelan *et al.*, 2001; Bailey *et al.*, 2002; Zhang and Gerstein, 2004). Therefore, reliable matching on a genomic scale requires additional steps, such as accumulating additional oligomer evidence beyond the first matching pair; monitoring when the number of candidate locations has been limited adequately; and sampling adaptively to extract information from different parts of the cDNA sequence, including the middle when necessary.

Approximate alignment

An approximate alignment step is necessitated by the large size of genomic segments, which makes a nucleotide-level alignment prohibitively time-consuming, and is therefore used in some form by virtually every cDNA–genomic alignment program. In EST_GENOME, approximate alignments are computed by using local Smith and Waterman (1981) alignments and the resulting segments are then recomputed with a global Needleman and Wunsch (1970) alignment. Spidey computes an alignment with increasing detail by performing successive BLAST runs at decreasing stringency levels.

In other programs, the predominant strategy has been a 'seed-and-extend' strategy, in which the program first finds significant oligomer matches between the cDNA and genomic segment, then extends these seeds to form longer matching fragments, and finally assembles a selection of these fragments into a collinear chain. The seed-and-extend strategy is found in a variety of programs, including those for genome–genome alignment (Chain *et al.*, 2003; Morgenstern, 1999; Batzoglou *et al.*, 2000; Kent and Zahler, 2000; Schwartz *et al.*, 2000; Ma *et al.*, 2002; Brudno *et al.*, 2003a,b; Bray *et al.*, 2003; Kalafus *et al.*, 2004), and constitutes the approach in several cDNA–genomic alignment programs. SIM4 finds matching seeds of 12-mers in the genomic segment, extends these seeds by nucleotide-level scoring of matches and mismatches, and then assembles the resulting 'exon cores' through dynamic programming (DP). MGAlign also applies DP, both to extend its fragments and to combine local alignments into longer ones. BLAT breaks the cDNA into 500-bp chunks, uses these chunks to create alignment fragments through a recursive seed-and-extend method, and then uses DP to stitch together these subalignments.

In contrast, GMAP uses an oligomer chaining method that involves neither seeds nor extensions. Rather, this method finds all matching 8-mers between the cDNA and genomic sequence, and then uses DP to find an optimal global chain of 8-mers. In this process, exons are not created explicitly, but instead emerge implicitly from the globally optimal distribution of 8-mer matches between the cDNA and genomic segment. Although exon–exon boundaries are defined only approximately by this method, their location is determined by both distant alignment information and local information. Oligomer chaining may extend an exon alignment that otherwise looks locally unfavorable, or terminate an exon alignment that otherwise looks locally favorable, when such decisions contribute toward a better global alignment. We have found that the use of global information is particularly important in the presence of sequence polymorphisms or errors, which can adversely affect local decision-making for extending fragments.

Splice site identification

Approximate alignment using 8-mers or other fragments generally does not have the resolution needed at the nucleotide level to detect splice sites accurately. To recognize splice sites correctly in the presence of sequence errors, a program must often introduce substitutions or gaps, shift nucleotides from one end of the intron to the other, or explore alternate locations for the splice sites.

Existing approaches to splice site identification are based upon two ideas. The first idea is to apply various heuristics to fix or adjust the approximate alignment to incorporate a splice site. For example, Spidey and MGAlign search for splice sites in the overlap between adjacent exons, and then trim the exons at the highest-scoring splice site, whereas SIM4 has an intron shifting procedure that adjusts the exon–exon junction to find the best pair of splice sites. The other idea is to use splice site models, such as scoring matrices (Salzberg, 1997; Brendel and Kleffe, 1998), which model the observed frequency of nucleotides near the 5′ and 3′ splice sites (Nakata *et al.*, 1985; Gelfand, 1989) and thereby provide clues about the presence and location of splice sites.

In contrast, GMAP handles this problem by using a formal DP procedure that we call sandwich DP. Sandwich DP involves two DP matrices, one for each end of an intron, and attempts to find the best alignment path across the diagonals of both matrices. Rather than

attempting to fix an existing approximate alignment, the method computes the whole subalignment in the region surrounding an intron. This approach guarantees that all possible combinations of substitutions, gaps and intron shifts are considered, and permits the use of various DP techniques. These techniques include specialized gap penalties that favor insertions or deletions of tri-nucleotides (Gotoh, 1999) and band-limited alignment (Sankoff and Kruskal, 1999), which enables efficient consideration of substitutions or gaps at a large distance away from the splice site.

Microexons

In addition to the above features, GMAP has an explicit procedure for detecting microexons and incorporating them into the alignment. Microexons as short as 1 nucleotide in length have found apparent experimental support (McAllister *et al.*, 1992; Sterner and Berget, 1993; Simpson *et al.*, 2000; Carlo *et al.*, 2000), and a computational study suggests that between 0.5 and 1.6% of mRNA sequences in various species contain microexons (Volfovsky *et al.*, 2003). Such short exons pose an acknowledged problem for cDNA–genomic alignment programs (Florea *et al.*, 1998). A procedure for identifying microexons has been developed by Volfovsky *et al.* (2003), and applied in a large-scale study. We further this work by integrating the detection procedure into the framework of a cDNA–genomic alignment program, and by adding a probabilistic extension that ensures that incorporated microexons are statistically significant.

ALGORITHM

In this section, we discuss the methods used by GMAP in the context of each of the major components needed for cDNA–genomic mapping and alignment. Specifically, we describe: (1) a minimal sampling strategy for genomic mapping, (2) oligomer chaining for generating approximate gene structures, (3) sandwich DP for identifying splice sites, and (4) microexon identification with statistical significance testing.

Minimal sampling strategy

For genomic mapping, GMAP uses a sampling strategy designed to minimize the number of oligomer lookups needed to map a cDNA reliably to the genome. Our minimal sampling strategy is based upon the use of long oligomers to achieve high specificity, combined with an adaptive sampling scheme to utilize mapping evidence from different parts of the cDNA sequence.

As discussed previously, the rationale for using long oligomers is their exponentially greater specificity in the genome, which means that mapping can be performed with few oligomer matches. Our choice of 24 as an oligomer length is guided by our own study of oligomer uniqueness in the human genome, as shown in Figure 1. This graph, based on the unmasked portion of the NCBI human genome (build 29), shows the percentage of the observed oligomers of various lengths that are unique in the genome. For example, among all 11-mers in the genome, only 0.1% of them have a unique position in the genome. Likewise, among all 14-mers, only 22.5% specify a unique position in the genome. On the other hand, when the oligomer length is 20 or more, the percentage of oligomers with a unique genomic location reaches an asymptotic level of 96–97%.

Our implementation of 24-mer lookups on a genomic scale requires some adaptation of the index table scheme of SSAHA (Ning
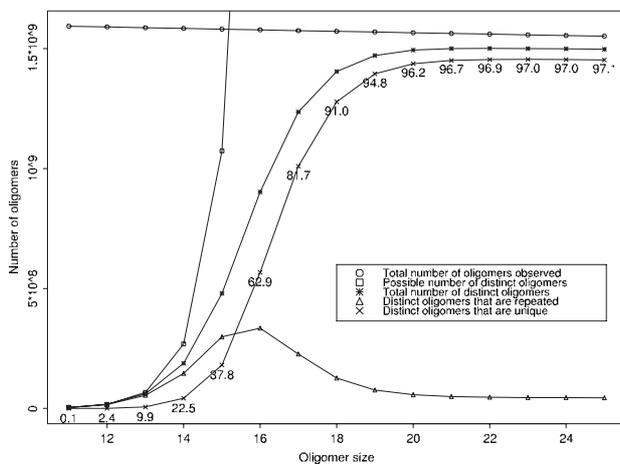
Fig. 1. Distribution of oligomers of various lengths in the masked region of the human genome (NCBI build 29). The horizontal axis represents various oligomer sizes from 11 to 25. The total space of possible oligomers increases exponentially, as shown by the exponentially increasing line. For each oligomer size, counts of all overlapping oligomers in the masked part of the human genome are shown by the top line, and the counts of distinct oligomers are shown by the topmost sigmoid line. Distinct oligomers can be divided into unique oligomers, which occur once (shown by the sigmoid line with percentages), and repeated oligomers, which occur more than once (shown by the bottom line).

et al., 2001). In that scheme, a position file contains the observed positions of oligomers in the genome, and an offset file contains pointers to the position file to indicate where a block of positions begins and ends for a given oligomer. Because this offset file contains an entry for each possible oligomer, its size grows exponentially with the oligomer length. In fact, 14-mers represent the current practical limit for the SSAHA data structure, because the corresponding index file occupies 1.1 GB. Extending this indexing scheme to 24-mers would yield a sparse offset file of $4^{24} = 281$ trillion 32-bit entries, which would be prohibitively large to store.

Therefore, in our initial implementation of GMAP, we tried a hashing scheme instead, where the space of 24-mers is mapped onto a space of 12-mers using a hash function. If a given 24-mer has a match somewhere in the genome, an entry for the 24-mer can be found in the expected hash bin. This entry then provides the appropriate offset into the position file.

Although this hashing scheme worked reasonably well, we subsequently found a more efficient solution by using a double lookup scheme, which breaks up the problem of finding a 24-mer into the problem of finding two 12-mers. In other words, we implement the SSAHA data structure for 12-mers, with the requirement that entries in the position table be pre-sorted in ascending numeric order within each oligomer. To find the positions for a given 24-mer, we look up two lists of genomic positions, one for the initial 12-mer and one for the terminal 12-mer. The desired set of 24-mer genomic locations is obtained by finding pairs of entries in these two lists that are separated by 12 nucleotides. The reason for pre-sorting the genomic positions within each oligomer is to make this procedure run in linear time with the number of genomic positions, rather than quadratic.

The size of the position file is determined by the genome size and by how often oligomers are sampled in the genome. Although minimal coverage of the genome can be achieved by sampling all

non-overlapping 12-mers in the genome, an overlapping sampling interval provides increased resolution, but at the cost of a larger position file. An additional advantage of overlapping sampling intervals in our scheme is that it permits lookups of oligomers other than 12-mers and 24-mers. For example, if we store 12-mers at an overlapping interval of 6 (which is our default), we can determine the genomic location of oligomers of length 12, 18, 24, and so on. These intermediate-length oligomers can be useful in genomic mapping. The use of 18-mers can give additional sensitivity for divergent sequences, such as in the cross-species genomic mapping of mouse cDNAs onto the human genome, and vice versa. In addition, short cDNA sequences often have too few 24-mers for reliable genomic mapping. In these cases, the program uses smaller oligomers: 18-mers if the cDNA is between 40 and 80 nt, and 12-mers if it is less than 40 nt.

In addition to using highly specific 24-mers, GMAP employs an adaptive sampling scheme designed to utilize mapping information from different parts of the cDNA sequence. The sampling process begins by scanning both ends of the cDNA sequence, and monitoring the results until a pair of 24-mers match to approximately the same location in the genome. The definition of 'same location' depends upon the length of the query cDNA, with an allowed genomic expansion of 1000 times the query length, subject to a default upper limit of 1 million nucleotides. Therefore, the program will not attempt to predict a long intron for a very short EST.

To avoid false localizations from a fortuitous pair of matches to the genome, the program continues to sample beyond the first pair of successful hits, in order to accumulate evidence of other possible localizations in the genome. This amount of further sampling is determined both by a minimum distance (default 48 nt) and by a minimum number of additional successful matches (default 3) required. If this process yields a limited number of genomic locations, the mapping process terminates.

On the other hand, if there are a large number of candidate genomic locations, then GMAP begins a sampling process that uses information from the middle of the cDNA sequence. This sampling process is performed iteratively, with the sampling interval halved in each round. At each sampling interval, the program looks for clusters on the genome with a high concentration of matches, with the provision that genomic positions be collinear with the cDNA positions. Sampling terminates when the correct genome location is resolved to a limited number of good candidates. This determination is made by setting a threshold at 70% of the number of matches in the best cluster, and requiring that only a limited number of clusters (currently defined as 10 or fewer) are above this threshold.

For each candidate cluster of 24-mers, the program extracts the corresponding segment from the genome, with the correct strand of the genome determined by the orientation of the matching 24-mers. To extend the genomic segment to regions that may be relevant for further alignment at the oligomer and nucleotide level, the program looks up the genomic positions of the nearest 12-mers that match to the ends of the cDNA sequence.

Oligomer chaining

For approximate alignment, oligomer chaining attempts to find a path of 8-mers that match between the cDNA sequence and each genomic segment found in the mapping step. The procedure is illustrated in the top part of Figure 2. Instead of the standard DP paradigm, which uses a matrix to align two sequences, oligomer chaining uses an
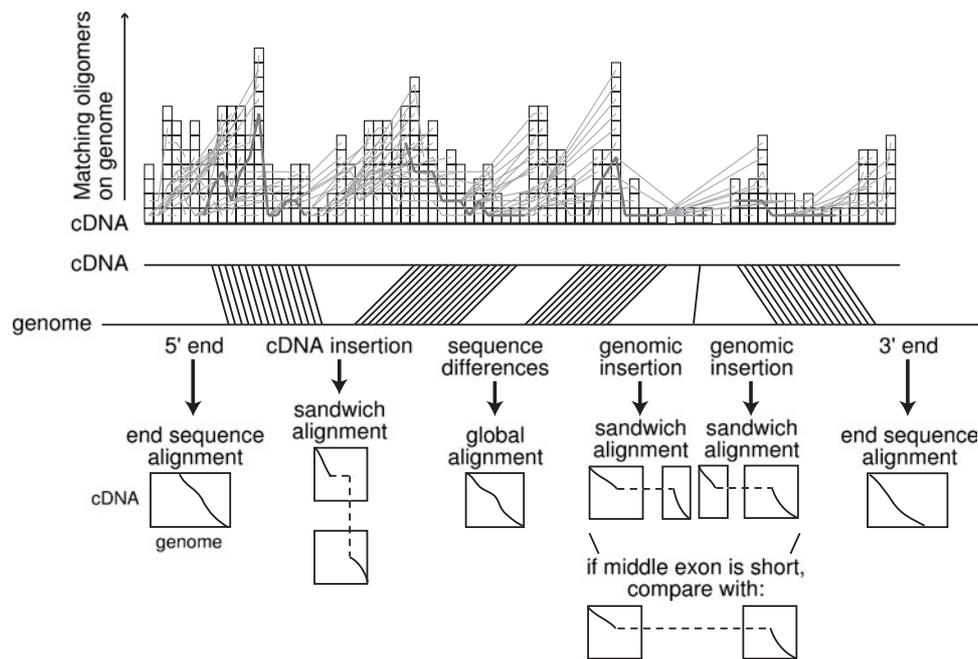
Fig. 2. Oligomer chaining and nucleotide-level alignment. The top part of the figure shows oligomer chaining. The horizontal axis represents positions on the cDNA sequence. Each cDNA position may have one or more matches of 8-mers to the genomic segment, represented by a vertical stack of cells. For each cell, the DP procedure looks for an optimal previous cell, as represented by thin diagonal lines between cells. The highest-scoring chain of 8-mer matches, represented by a thick line, describes the optimal approximate alignment. This alignment may contain jumps in cDNA or genomic coordinates, due to introns, cDNA insertions or sequence differences. These jumps are resolved by various nucleotide-level alignment procedures, represented in the bottom of the figure by various DP matrices. Sandwich alignments bridge large coordinate jumps across introns (horizontal dashed line) or long cDNA insertions (vertical dashed line). The existence of short exons is resolved by an exon testing procedure that compares alignments with and without the short exon.

equivalent but more efficient representation in the form of an array of linked lists. Each position in the array corresponds to an overlapping 8-mer in the cDNA sequence, and each 8-mer has a linked list of positions in the genomic segment where that 8-mer is found. These linked lists are represented in the figure as a vertical stack of cells at each cDNA position. Each cell also contains placeholders for the optimal subscore to that point and for a pointer to the best previous cell that produced the optimal subscore.

The array of linked lists is generated by first pre-scanning the cDNA for overlapping 8-mers and noting which 8-mers are present, and hence relevant. This pre-scan prevents unnecessary work later, because most of the 8-mers in the longer genomic sequence are irrelevant. Then the algorithm scans the genomic segment for relevant 8-mers and adds their genomic positions to a list maintained for each relevant 8-mer. Finally, the algorithm scans the cDNA again, making a copy of the appropriate position list for each element of the array.

After building this data structure, oligomer chaining proceeds with a DP procedure that assigns a subscore and pointer to each cell, starting from the beginning of the cDNA sequence. For each cell, the algorithm looks backward to cells at previous cDNA positions to identify the cell that both is consistent and generates a maximal score to the given cell. A previous cell is consistent if its genomic position is lower than that of the given cell, which enforces collinearity of the cDNA and genomic sequences. The score for the cell is the score of the previous cell plus 1 to indicate the length of the chain. Because introns will cause 8-mers in the cDNA not to match, the algorithm compensates for such cases by adding enough points

to ensure that local extension does not gain an unwarranted advantage over an intron.

One cost of our approach is greater computational complexity than one based on larger fragments. As described so far, oligomer chaining is $O(m^2 g^2)$, where m is the length of the cDNA and g is the average number of cells per linked list, which is generally proportional to the length of the genomic segment. (A total of mg cells must be processed, and at each cell the algorithm must look back at the previous set of cells processed.) In order to reduce the complexity to $O(mg^2)$, we impose a sufficiency limit on the look backward. Note that this limit applies only to the cDNA sequence coordinates; there is no limitation on the look backward in genomic sequence coordinates. The sufficiency limit has a default value of 60, which expresses our calculated expectation that we should find at least one matching 8-mer between the cDNA and genome within that distance, even accounting for extremely low sequence quality. By using probability calculations based on finite-state automata (Atteson, 1998), we estimate that if the sequence error rate is 5%, then the chance of failing to have an error-free stretch of 8 nucleotides out of 60 total nucleotides is 3.8×10^{-6}.

The pointer and optimal subscore for a given cell are based on the best solution found within this sufficiency limit. However, a cDNA sequence may have a local concentration of mismatches or gaps that precludes 8-mers from being identified in a particular stretch. Therefore, if no matching 8-mer is found within the sufficiency limit, the algorithm will continue looking backward as far as needed to find a match. This provision allows the algorithm to
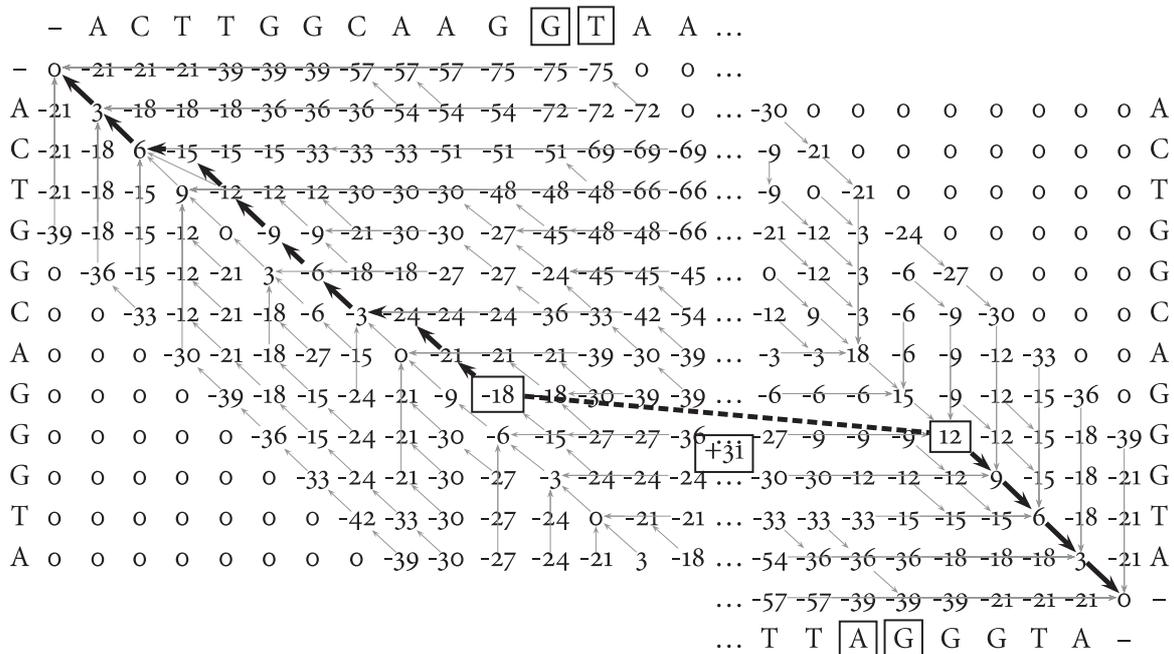
Fig. 3. Sandwich DP for identifying splice site boundaries. This solution corresponds to an intron in EST sequence BF846255, shown in the middle of Figure 4. Two alignment matrices are shown. The cDNA sequence is shown on the common vertical axis, and the 5′ and 3′ genomic ends of the intron on the horizontal axis of each matrix. The two matrices are solved 'outside in', as shown by the direction of the arrows. The optimal solution, shown in bold, is found by adding terms in adjacent rows, plus a reward for canonical introns, as indicated by the boxed GT–AG pair.

cope with sections of cDNA that have extremely poor sequence quality.

In order to reduce the complexity further to $O(mg)$, we note that one cell in the linked list for a given 8-mer usually has a score that dominates over the scores of other cells in the list. Domination occurs if the best score exceeds the second best score by more than the intron compensation discussed previously. In such cases, the dominating cell can be marked by a pointer, so that downstream cells looking backward to the given 8-mer need consider only that cell.

Finally, the overall approximate alignment is obtained from the optimal path of cells, which represents a set of matches between 8-mers in the cDNA and genomic segment. This path of 8-mers is converted into an alignment at the nucleotide level, using a linked list representation, in preparation for alignment procedures at the nucleotide level.

At this point in the algorithm, the program can assess the quality of the cDNA against the genome, based on the number of short breaks in the alignment. This quality information can be useful in guiding the rest of the algorithm. The fraction of such short breaks relative to the total alignment length is defined to be the defect rate, and is used to classify the cDNA sequence as being of high (defect rate <0.3%), medium (0.3–1.4%), or low quality (>1.4%). This classification enables appropriate parameters for nucleotide-level alignment to be selected automatically, so that substitutions and gaps are more likely to be introduced for low-quality sequences, and less likely for high-quality sequences.

Sandwich DP and other nucleotide-level alignment

GMAP uses a procedure we call 'sandwich DP' to compute sub-alignments around introns. Actually, sandwich DP can be used to

handle not only introns, but also long cDNA insertions relative to the genome, which occur rarely. For introns, the jump in genomic coordinates is much greater than that for cDNA coordinates; for cDNA insertions, the opposite is true. For simplicity, we describe primarily the intron case, as shown in Figure 3, in which the cDNA sequence is placed on the common vertical axis and the genomic ends of the intron are placed on the horizontal axis. To handle cDNA insertions, the procedure switches the assignments of cDNA and genomic sequences to the two axes.

In sandwich DP, the goal is to find an optimal path from the upper left corner to the lower right corner. For the intron case, this path bridges the coordinates for the cDNA sequence but allows genomic coordinates to jump across the intron. To find the optimal path, each matrix is scored 'outside in' by the usual Needleman and Wunsch (1970) procedure, which enforces an alignment to the ends of the intron. However, once scoring is complete, we cannot proceed directly to backtracking, because a single optimal score is not directly available. Rather, we must find the optimal combination of scores between the two matrices by evaluating adjacent rows (representing adjacent cDNA positions) and pairs of columns within those rows. Testing each adjacent pair of rows is equivalent to shifting nucleotides across the gap, whereas selecting different columns is equivalent to trying different splice sites. The algorithm selects the combination that produces the maximum combined score, including a reward if the solution results in a canonical or semi-canonical splice site.

Sandwich DP is one of several nucleotide-level alignment procedures used to fill in gaps in the approximate alignment. In addition to introns, other types of sequence differences can cause 8-mers not to align in the oligomer chaining procedure and thereby yield discontinuities or jumps in the cDNA or genomic coordinates in the

alignment. Each of these types of coordinate jumps is handled by an appropriate nucleotide-level procedure, as shown in the bottom part of Figure 2. These procedures are applied in a particular order in four passes through the alignment.

In the first pass, the algorithm solves regions where the cDNA and genomic coordinate jumps are approximately equal, indicating the presence of small sequence differences such as mismatches or short insertions or deletions. The program fills in these gaps with global Needleman–Wunsch DP, which enforces alignment to both ends.

In pass 2, the algorithm validates the existence of short exons, defined as those with fewer than 80 nt (but at least 8 nt, which is the minimum resolution of oligomer chaining). This step is necessary because an approximate alignment from oligomer chaining can contain small islands of 8-mers, as small as a single isolated 8-mer, which may represent either a true short exon or a spurious match. If the match is spurious, a better alignment should result by splitting the short exon and merging the halves into adjacent exons. Therefore, to decide whether a short exon does indeed exist, the algorithm attempts to align the region under the two assumptions that the short exon is present (meaning two introns and a middle exon) or that it is absent (meaning one intron). It then merges in the subalignment that provides the better alignment score.

In pass 3, the algorithm fills in large relative jumps in coordinates, where the jump in genomic coordinates is much greater than that of the cDNA jump, or vice versa. The former situation is due to introns, and the latter, which occurs rarely, is due to long cDNA insertions. The algorithm handles these jumps by applying the sandwich DP procedure described previously.

In the fourth and final pass, the algorithm extends the 5' and 3' ends of the cDNA sequence, by using DP for the sequence ends. End sequence alignments are computed by constraining one end of the alignment and allowing the distal end to terminate at an optimal stopping point. This procedure is implemented by a modified Smith and Waterman (1981) local alignment, in which we choose an optimal score from anywhere in the matrix for backtracking, but do not reset negative scores to zero during the scoring procedure. If all scores in the matrix are negative, the end is not extended.

Each of the above DP procedures employs a band-limited search through the score matrix (Sankoff and Kruskal, 1999). Such an approach is relatively sound because oligomer chaining bounds the solution well from a global perspective, leaving only small sequence edits to be performed. Another implementational detail is that before each nucleotide-level DP procedure is performed, some of the nucleotide matches on each end of the coordinate jump must be 'undone' or 'peeled back'. The resulting margin gives the nucleotide-level DP procedure freedom to find a better alignment than that found by the coarser oligomer chaining procedure.

Our DP procedures allow us to handle codon insertions and deletions gracefully by an appropriate gap penalty function. We use a 'step function' gap penalty, where instead of a per-nucleotide extension as in the usual affine gap penalty, we have a per-codon extension penalty. This per-codon penalty is equal for gaps of 1, 2 and 3 nucleotides, likewise for 4, 5 and 6 nucleotides, and so on. As a result, our algorithm has a preference for insertions and deletions that are multiples of 3. Similar gap penalties that favor multiples of 3 have been used in other programs (Gotoh, 1999). The preference for trinucleotide gaps reflects selection pressure at the protein level to avoid frameshifts and preserve the coding region.

The above nucleotide-level procedures are tried under the two assumptions that the cDNA sequence is sense or antisense, and the cDNA direction is determined, if possible, based on the higher alignment score in terms of canonical splice sites, matches, substitutions and gaps. When multiple candidate alignments are found, due to multiple genomic segments found in the mapping step, the candidates are ranked and reported according to their alignment score.

Probabilistic microexon identification

The problem of detecting microexons is challenging because merely changing parameters to identify them often backfires, resulting in spurious extra exons in the middle of introns. The problem is particularly acute for long introns, which have a greater opportunity to have an exact match by chance to a given short oligomer. In such cases, a program must decide whether extra nucleotides in the cDNA are due to a microexon or to an insertion in the adjoining exons.

GMAP has an explicit procedure for finding microexons, based on the method by Volfovsky *et al.* (2003). It applies this procedure in pass 3 of nucleotide-level alignment when the initial alignment of an intron is neither canonical nor semi-canonical, and when the alignment surrounding the intron has more than an acceptable number of mismatches or gaps (0 for a high-quality sequence, 2 for medium and 3 for low). Also, we require that a microexon be reported only if it matches perfectly to the genomic sequence and is surrounded by two canonical introns.

When these conditions are met, the program calculates a lower bound on the microexon length that satisfies a given statistical significance level ($p < 0.01$ by default). The calculation imposes a higher minimum length requirement for a microexon in a longer intron, to offset its higher likelihood of an exact match by chance.

We assume a simple model where nucleotides in an intron of length L are generated independently with uniform distributions of 1/4 probability per base. For a microexon with e nucleotides, the probability p that the microexon matches somewhere in the intron and is surrounded by two canonical introns is

$$p = 1 - \left[1.0 - (1/4)^m\right]^L \qquad (1)$$

Note that here $m = e + 8$ to include the exon length e as well as the 8 positions for the two required canonical dinucleotide pairs. If we solve this equation for m, we obtain

$$m = -\frac{\log(1 - (1 - p)^{1/L})}{\log(4)} \qquad (2)$$

Therefore, given an intron length L and an upper limit on the statistical significance p, GMAP calculates the lower limit for m, and then searches for microexons that are of size $e = m - 8$ or longer.

Like the Volfovsky method, our procedure searches for GT and AG pairs in the 5' and 3' ends surrounding the intron, but considers only those that satisfy the calculated lower bound on the microexon length. Also, our procedure looks only within 12 nt of the alignment boundaries rather than the 30 nt by Volfovsky, because longer microexons would have been identified by oligomer chaining. Potential microexons are then scanned across the intron using Boyer and Moore (1977) sublinear-time string matching, and accepted if they are surrounded by the requisite AG and GT dinucleotide pairs.

Similarly, in pass 4, GMAP can find statistically significant microexons at the 5' and 3' ends of the alignment. GMAP is very conservative in applying this procedure, requiring a high-quality

	1 sequence difference BF669985:519..530, chr +X	2 sequence differences BF846255:69..80, chr +9	0 sequence differences BF591480:252..264, chr +11
Evidence	S36950, M64241, M73791	No overlapping ESTs	AI051280, BE858123

GMAP

```
CAGAAGGTATGT...CCTTAGATCCACA      ACTTGGCAAGGTAAAT...ATTTAGGGTA      ACATTGTGAAGT...TTGTTTGGTGAC
||||||>>>>>>...>>>>>>|||-|||      ||-||||-||>>>>>>...>>>>>>||||      ||||||======...======||||||
CAGAAG      75      ATC ACA      AC TGGC AG    7812      GGTA      ACATTG      98      GGTGAC
```

BLAT

```
CAGAAGGTATGTAG...TAGATCCACA       ACTTGGCAAGGTA...TCATTTAGGGTA       Same as GMAP
||||||  |======...======||||      |||-|||======   ======|||||
CAGAAGAT      76      CACA        ACT GGC    7813      AGGGTA
```

dds/gap2

```
Same as GMAP                      ACTTGGCAAGGTAA...CATTTAGGGTA       Same as GMAP
                                  |||-||||======...======|||||
                                  ACT GGCA    7813      GGGTA
```

GeneSeqer

```
Same as GMAP                      Misses 5′ exon                    Same as GMAP
```

MGAlign

```
Same as GMAP                      ACTTGGCAAGGTAAAT...TTTAGGGTA       ACATTGTGAAG...GTTGTTTGGTGAC
                                  |||-||||  |======...======|||      |||||======...======  ||||||
                                  ACT GGCAGG    7813      GTA        ACATT      98      GGGTGAC
```

SIM4

```
Same as GMAP                      Same as GMAP                      ACATTGTG...AAGTTGTTTGGTGAC
                                                                    ||||||>>>...>>>--|---||||||
                                                                    ACATT    92      G    GGTGAC
```

Spidey

```
CAGAAGGTATGTAGTG...GATCCACA       ACTTGGCAAGG...GTCATTTAGGGTA       Same as GMAP
||||||  |  ======    ======||     |||  |======...======  ||||||
CAGAAGATCA      76      CA         ACTGG    7814      CAGGGTA
```

Fig. 4. Splicing errors. This figure shows alignments generated by various programs around introns in three sequences. Alignments have been formatted in a uniform style. The first column shows a canonical intron (marked by '>') from an EST with one sequence difference nearby: a single gap (marked by '−'). The second column shows a canonical intron from an EST with two gaps nearby. The third column shows a non-canonical intron (marked by '=') with no mismatches or gaps. Alignments have the genome sequence on top and the cDNA on the bottom, with the cDNA in its forward direction. Numbers below each intron indicate its length in nucleotides. Other overlapping ESTs are listed as evidence for the alignment found by GMAP.

sequence, an adjacent canonical intron, and the remaining subsequence to match exactly to the genome. When these conditions are met, the program tests each candidate microexon of length e in the remaining end sequence from longest to shortest. For each microexon length, the program computes the maximum length L of genomic sequence for the microexon to be statistically significant:

$$L = \frac{\log(1 - p)}{\log(1 - (1/4)^m)} \tag{3}$$

In this case, the number of matches is $m = e + 4$, to account for the microexon length e and the canonical dinucleotide pair in the intron. This amount of genomic sequence is then scanned for an exact match of the microexon using a Boyer–Moore search, and the microexon is accepted if it yields a canonical intron.

RESULTS

Examples

The methods employed by GMAP enable it to handle certain types of alignment problems that pose challenges for existing programs. Some illustrative examples of these problems are shown in Figures 4 and 5.

Figure 4 shows some cases of splice site detection in the presence of sequence error. For the first EST, which has one sequence difference relative to the genome, the canonical intron is recognized by five out of seven programs. However, for the second EST, which has two sequence differences, only GMAP and SIM4 can recognize the canonical intron. Programs can be overly liberal in identifying introns as being canonical, thereby resulting in false positives. This is seen as the third EST, in which SIM4 appears to overcall a canonical intron by introducing gaps of 5 nt in an mRNA that otherwise has perfect sequence identity to the genome.

Another class of errors seen in cDNA–genomic alignment involves gene structure, manifesting as missing or extra exons, as illustrated in Figure 5. The first example shows an apparent 6-nt difference between the cDNA and the genome. GMAP and GeneSeqer interpret this as a microexon surrounded by two canonical introns. Other programs give less plausible alignments, involving a combination of non-canonical introns, 4-nt microexons, and nucleotide substitutions and gaps. The second example in Figure 5 shows that many alignment programs truncate their alignment prematurely in the presence of substitutions or gaps. In this example, GMAP and GeneSeqer are able to extend the alignment, thereby revealing a canonical intron and an additional exon. The third example in Figure 5 shows how initial and terminal exons can be difficult for some alignment programs to find. This EST has a final 31-nt exon that is missed by various programs, which try instead to extend the alignment locally. Although one must generally be cautious in making inferences from ESTs with sequence errors, analysis of a rare gene of interest may depend on maximizing information from a single EST. In these examples, the predicted exons are indeed supported by other sequences, as listed in the figure.

	Microexon BM467080:555..571, chr +3	Low quality 3′ end BG118317:738..757, chr −12	Final exon AA036958:424..439, chr −1
Evidence	BE261245, BQ879950, CK004314	BG385668, BG747641, BM745724	CD612209, BF382871
GMAP	`GAGTCAGGTA...CAGCATCAGGTA...TAGACAA` `|||||||>>>...>>>|||||||>>>...>>>||||` `GAGTCAG  393  CATCAG  198  ACAA`	`GCTATATGAAGAGGTA...CAGGAGATCCGG` `|	||||-||||>>>...>>>||-|||||||` `GTTGTATG AGAG 522 GA ATCCGG`	`AAGGTA...CAGGCTGATTCACCCC` `|||>>>...>>>|||||||||||||` `AAG  6772  GCTGATTCACCCC`

BLAT	`GAGTCAGGTA...CTTCATCTCA...TGTAGACAA`
	`|||||||===...===|||||===...===||||||`
	`GAGTCAG  48  CATC  543  AGACAA`
		Alignment ends at nt 651	`AGGAAGG`
			`|||||||`
			`AGGAAGG`

dds/gap2	`GAGTCAGGTA...CCTCATGTAGACAA` `|||||||===...===||| -||||||` `GAGTCAG  590  CATC AGACAA`	`GCTATATGAAGAGGTATGTT...GATCCGG` `|	|||||-|||| ||===...===||||` `GTTGTATGA GAGGAAT 523 CCGG`	`AAGGT  ATTTGTCCC` `||||  --|||   |||` `AAGGCTGATTCACCCC`
GeneSeqer	Same as GMAP	Same as GMAP	Same as GMAP

MGAlign	`TCAG  595  AGACAA`
	`||||###...###||||||`
	`TCAG  4  AGACAA`
		Alignment skips nt 691 to 748	Same as GMAP

SIM4	`GAGTCAGGTA...CTGCCCTCATGTAGACAA`
	`|||||||===...===----|||- ||||||`
	`GAGTCAG  586  CAT CAGACAA`
		Alignment ends at nt 714	`AAGG T ATTGTCCC`
			`||||-|-|| |||`
			`AAGGCTGATCACCCC`

Spidey	`GAGTCAGGTACTTGAA...TAGACAA` `|||||||	===...===||||` `GAGTCAGCATCAG 591 ACAA`	Alignment ends at nt 705	Same as GMAP

Fig. 5. Gene structure errors. This figure shows alignments for three additional ESTs. Notation follows that of Figure 4, with the addition of the character '#', which indicates a dual break in the alignment in both the genome and the cDNA sequence. The lengths of the two breaks are indicated above and below the alignment.

Experiment 1: Human mRNAs

We performed a comparison of GMAP with several existing genomic alignment programs on full-length human mRNAs. For this analysis, we used the 1 November 2004 release of Ensembl mRNAs, and extracted the 885 sequences annotated to be on chromosome 22. We ultimately excluded two sequences from this set, because subsequent runs of both GMAP and BLAT failed to map them to chromosome 22. Sequence ENST0355936 was placed by GMAP on chromosome 2 and by BLAT on chromosome 7, and sequence ENST0357004 was placed by GMAP on chromosome 1 and was not localized by BLAT to anywhere on the human genome.

The Ensembl data set contains annotated exon boundaries, which we used as a gold standard. Our data set contained a total of 8634 exons. Some exons were extremely short, with 41 exons having lengths of 3–10 nt. In addition, some inter-exon regions were also extremely short, with 125 having lengths of 1–7 nt. These regions between exons are annotated as 'introns', although some programs may annotate these simply as small cDNA deletions. In fact, experimental evidence (Wieringa *et al.*, 1984) suggests that introns require at least 70 or so nucleotides for splicing to occur, and computational evidence (Yu *et al.*, 2002) provides evidence for a species-specific minimal intron length. For 16 of the 41 short exons, there was a short intron immediately preceding or following. Further inspection of these 16 short intron/exon patterns and comparison of the alignments with available EST evidence suggests that these patterns may have been introduced computationally in order to maintain the reading frame. The remaining 25 short exons surrounded by introns of typical lengths may be considered to be true microexons.

To test how robust alignment programs are to sequence error, we generated two additional test sets by computationally introducing random mutations at rates of 1 and 3%. A similar mutation paradigm has been used to evaluate *ab initio* gene structure prediction programs (Burset and Guigó, 1996). For each position in an mRNA sequence, we generated a random number that determined, with 1 or 3% probability, whether a mutation would be introduced at that position. If a mutation event was selected, we generated an additional random number that determined whether the mutation was a substitution, insertion or deletion, with 80, 10 and 10% probabilities, respectively. These probabilities are the same as those used by Tammi *et al.* (2003) in their simulation of observed errors in shotgun sequences. For substitution and insertion events, we generated a nucleotide randomly, without regard to the original nucleotide. Therefore, the original nucleotide may have been resubstituted in the given position, resulting in no change.

We provided each of the three mRNA data sets as input to the following programs that were available to us and which were designed to run primarily on vertebrate mRNAs: BLAT version 31 (31 October 2004), dds/gap2 version 30 October 2003, MGAlign version 1.3.7 (25 September 2003), SIM4 version 21 September 2003, Spidey version 1.35 and GMAP. Parameters used were all default, without any additional flags, with the following exceptions: for SIM4, we used the flags 'A = 4 P = 1', which prints the alignment and removes poly-A tails (which are not present in these data sets anyway); for Spidey, we used the flag '-p 0', which prints the summary and alignment; and for GMAP, we used the flags '-BA' to indicate a batch run that pre-loads genomic files into RAM and prints the alignment. For dds/gap2, we ran the three programs dds, ext and gap2, each without

Table 1. Results of aligning 883 Ensembl mRNAs from chromosome 22

Condition	Time	Gene structure errors							Splicing errors				Union
		Mis5	Mis3	MisM	MisI	Extr	Mult	Total	Shift	Over	Mult	Total	
GMAP, 0%	1:12	1	0	0	0	0	1	2	0	0	0	0	0
GMAP, 1%	1:20	3	6	3	0	2	1	15	0	0	0	0	15
GMAP, 3%	1:30	10	7	8	3	2	2	32	5	1	0	6	34
BLAT, 0%	6:40	20	9	8	1	1	2	41	4	0	6	10	47
BLAT, 1%	6:34	21	10	8	5	24	7	75	71	1	13	85	141
BLAT, 3%	6:29	23	12	6	12	99	32	184	172	0	54	226	331
dds/gap2, 0%	8:25:54	3	9	20	17	6	22	77	47	0	10	57	93
dds/gap2, 1%	8:23:21	4	10	20	14	5	20	73	86	0	17	103	135
dds/gap2, 3%	8:01:09	1	12	21	14	6	20	74	163	0	40	203	227
MGAlign, 0%	2:45	1	0	13	2	5	4	25	31	1	3	35	45
MGAlign, 1%	1:26:24	2	0	13	5	6	6	32	132	1	43	176	188
MGAlign, 3%	2:47:01	4	1	12	13	26	10	66	213	1	160	374	400
SIM4, 0%	:38	4	0	18	20	1	10	53	37	1	1	39	64
SIM4, 1%	:41	6	1	18	22	2	12	61	42	2	1	45	74
SIM4, 3%	:39	12	1	18	23	2	11	67	52	2	2	56	89
Spidey, 0%	1:26	16	2	21	11	2	18	70	54	0	36	90	112
Spidey, 1%	1:24	15	2	21	11	1	14	64	196	0	81	277	299
Spidey, 3%	1:29	13	3	19	11	5	19	70	227	1	268	496	517

Entries indicate the number of sequences with errors of various types. Key: Mis5, Mis3, MisM and MisI = missing 5′, 3′, microexons, and other internal exons; Extr = extra exon; Shift = shifted canonical intron to another genomic position; Over = overcalled canonical intron; Mult = multiple errors of a given class. The final column shows the union of all sequences with some error. Run times are in (hours:minutes:seconds) are for mapping and alignment by GMAP and BLAT, and for alignment by the other programs.

any additional flags. For pure alignment programs, we provided the genomic segment corresponding to each mRNA, with an additional 1000 nt on each end. We tested all programs on an Intel Linux machine with 2 Xeon processors at 2.4 GHz with 2 GB of RAM running RedHat Linux.

For each data set, we developed a gold standard set of exon–exon boundaries in the cDNA. The gold standard for the unmutated data set was derived from the exon coordinates provided by Ensembl. Annotations were added to indicate whether the corresponding introns had a canonical or non-canonical pair of dinucleotides, and to mark short exons (10 or fewer nucleotides) and short introns (7 or fewer nucleotides). Gold standards for the mutated data sets were computed by shifting the exon–exon boundaries accordingly when they followed insertions or deletions.

We parsed the output of the different programs into a uniform format that contained the computed exon boundaries, plus the dinucleotide pair for each intron. We then compared the computed exon boundaries with the gold standard to count errors of various types. Our comparison involved a DP procedure to find corresponding exon–exon boundaries between the computed and gold standard gene structures. This procedure was relatively simple to implement, but was needed to score results for the mutated data sets, which caused programs to frequently miss exons or include extra ones, and to shift splice sites by various distances. Complete input and output files for this and the other experiments are available as Supplementary Material.

We classified errors into two classes—gene structure errors and splicing errors—and counted the number of mRNAs for which an error occurred. Counting on a per-sequence basis makes sense because splicing and gene structure decisions for a sequence are often interrelated, making it difficult to assess how many individual errors in a sequence were actually committed. Hence, a sequence that had more than one error of a given class was counted as a single sequence error, with such cases being placed into a 'multiple error' category. Gene structure errors occurred in cases where the genomic alignment program missed one or more 5′, 3′ or internal exons (either microexons or longer ones), or inserted an extra exon. Splicing errors were counted when a program shifted either end of a gold standard canonical intron to a different genomic position, or when it shifted either end of a gold standard non-canonical intron to create a canonical intron. We call the latter error 'overcalling' a canonical intron. In the gold standard, we found 2 non-canonical introns that could be converted to a canonical one with 0 substitutions or gaps; 38 that could be converted with 1 substitution or gap; 6 with 2 substitutions or gaps; 11 with 3 substitutions or gaps; and 3 with 4 substitutions or gaps. We excluded these introns from being counted as either shifting or overcalling errors, since many programs are designed to convert these non-canonical introns into canonical ones. In particular, on the unmutated data set, SIM4 converts all of the above non-canonical introns into canonical ones, and GMAP converts all that involve 0, 1 or 2 substitutions or gaps, or that involve 3 contiguous gaps.

Because we evaluated gene structure and splicing errors separately, a sequence could have been counted as an error in each class, which occurred especially when the errors were interrelated. For example, failure to recognize an internal exon, especially a microexon, can lead to an error in finding the correct splice site. (On the other hand, failure to recognize a 5′ or 3′ exon would not lead to a splicing error, since no intron would have been predicted.) Because the two error classes are not mutually exclusive, we also tallied the union of sequences with one or more errors of any type.

The results of this experiment are shown in Table 1. On the unmutated data set, GMAP made no errors in identifying splice sites. In

terms of gene structure, it had two differences from the gold standard, for a per-sequence error rate of 0.2%. However, in these two cases, it is not clear whether the gold standard or GMAP has the more plausible alignment. On sequence ENST0354373, GMAP starts the alignment at position 13, rather than creating an initial exon of 13 nt followed by a non-canonical intron. On sequence ENST0338911, GMAP aligns an initial exon of length 120 with 18 substitutions and 4 gaps, instead of creating three exons of lengths 40, 15 and 65, separated by two non-canonical introns. In terms of microexons, GMAP identified all 25 microexons in the gold standard that were not adjacent to a short intron.

Other alignment programs had higher error rates than GMAP on the unmutated data set. In identifying gene structure, MGAlign came closest with 25 wrong sequences (2.8% error rate). Error rates for the remaining programs ranged from 4.6 to 8.7%. In identifying splice sites, BLAT had 10 errors (1.1% error rate), while the other programs had error rates between 4.0 and 10.2%. Interestingly, 9 out of the 10 BLAT splicing errors were associated with microexons, because they were missed, predicted with the wrong length, or matched to the wrong place in the intron. Overall, if we consider the union of sequences with gene structure errors and splicing errors, GMAP had no errors, whereas the next best performer had an error rate of 5%.

On the mutated data sets, GMAP also outperformed other programs. In terms of gene structure errors, on the 1% data set, GMAP made errors in 15 sequences (1.7% error rate), while the other programs had error rates of 3.6–8.5%. On the 3% data set, GMAP made gene structure errors on 32 sequences (3.6% error rate), while the other programs had error rates of 7.5–20.8%. The gene structure errors made by GMAP typically involved short exons, including microexons and short missing 5′ and 3′ exons. For example, in the 3% data set, the missing end exons were all less than 25 nt.

In terms of splicing errors, GMAP outperformed the other programs by even larger margins. On the 1% data set, GMAP made no errors, while the other programs had error rates of 5.1–31.4%. On the 3% data set, GMAP made errors in 6 sequences for an error rate of 0.7%. By comparison, other programs had error rates of 6.3–56.7%, due predominantly to shifted canonical splice sites.

Running times for the different programs are also shown in Table 1. The running time for GMAP was for a single thread. The running time for BLAT is shown for client–server mode. Times for GMAP and BLAT do not include startup time for the server or for memory mapping of the oligomer index files. (GMAP requires about 3 min to memory map files for the human genome on its first run, but much less time on subsequent runs if pages from the file are still resident in memory. BLAT requires a somewhat longer time to start its server.) Running times for the remaining programs measure cDNA alignment to their corresponding genomic segments, and include the time needed to restart the program for each alignment. The running time for dds/gap2 is extremely long, which probably reflects its reliance upon alignment procedures at the nucleotide level. We also note that MGAlign shows a substantial increase in running time with the mutated data sets, perhaps reflecting some underlying characteristic of its handling of substitutions and gaps.

Experiment 2: Human ESTs

Our second experiment assessed the quality of genomic mapping and genomic alignment on ESTs. We compared GMAP with BLAT, the only other integrated program for mapping and alignment available to us. We constructed a test set of 48,441 ESTs by taking every 100th human sequence from GenBank. We used each program to map these ESTs onto the NCBI human genome version 35, ignoring contigs that were labeled as unmapped. We ran GMAP in batch mode and the server version of BLAT version 31 on the Linux Intel Xeon platform described previously. Run time for the test set was 3 h and 2 min for BLAT and 32 min for GMAP.

Because ESTs are of widely differing quality, we assigned each EST a quality score, which was the percentage identity of the EST relative to the genome as determined by the higher identity score between the GMAP and BLAT alignments.

The results of our comparison are shown in Figure 6. The top plot shows the number of ESTs at each quality level. There were 1083 ESTs that neither program could align to the genome. In addition, there were 3472 ESTs (or 7%) that had 60% identity or less by both programs. These ESTs are shown as the leftmost vertical bar in the top graph, and were excluded from further analysis. Approximately half (20,935 or 47.7%) of the remaining ESTs had quality scores of 98% or more.

For each EST, we determined whether GMAP or BLAT provided a better alignment. Because the two programs report scores differently, we scored all alignments using the BLAST scoring system (Altschul *et al.*, 1990), which assigns +1 point for matches, −3 for mismatches, −5 for gap openings and −2 for gap extensions, including the first nucleotide in the gap. Because the PSL output of BLAT includes introns in its count of genomic gaps, we ignored any genomic gap >10 nt as a putative intron in computing the BLAST alignment score for that program. To disregard minor differences between alignments, if the difference between alignment scores was 10 points or less, we considered the alignments to be a tie.

In comparing the ESTs, there were three possibilities to consider: (1) both programs aligned the EST to the same (overlapping) genomic location, (2) the programs aligned them to different locations, and (3) only one program provided an alignment. The first category was represented by 43,407 ESTs (96.5%); the second by 1206 (2.7%); and the third by 356 (0.8%).

Among the 43,407 overlapping cases, 32,187 (or 74.2%) ESTs had a tie score; 8032 cases (18.5%) had a better alignment by GMAP; and 3188 cases (7.3%) had a better alignment by BLAT. The middle graph of the figure shows the counts of ESTs for which the alignment was superior by either program, distributed according to their quality score. The graph shows that below a quality score of 85%, alignment quality was evenly divided between GMAP and BLAT. However, above a quality score of 85%, GMAP provided a better alignment more often than BLAT. If we consider the 20,635 ESTs with 98% identity or more, 18,363 (89.0%) were ties, 1953 (9.5%) favored GMAP, and 319 (1.5%) favored BLAT.

The bottom graph of Figure 6 shows the non-overlapping cases, which include the 1206 ESTs aligned to different genomic locations and the 356 aligned by only one program. These cases represent a relatively small percentage of the ESTs, but as before, above a quality score of 85%, GMAP provides a better alignment more often than BLAT does.

Experiment 3: *Arabidopsis* mRNAs

Observations of nucleotide frequencies around splice sites indicate that they are species-specific (Senapathy *et al.*, 1990). In addition, intron lengths have significantly different distributions in different species, with *Caenorhabditis elegans, Drosophila melanogaster* and
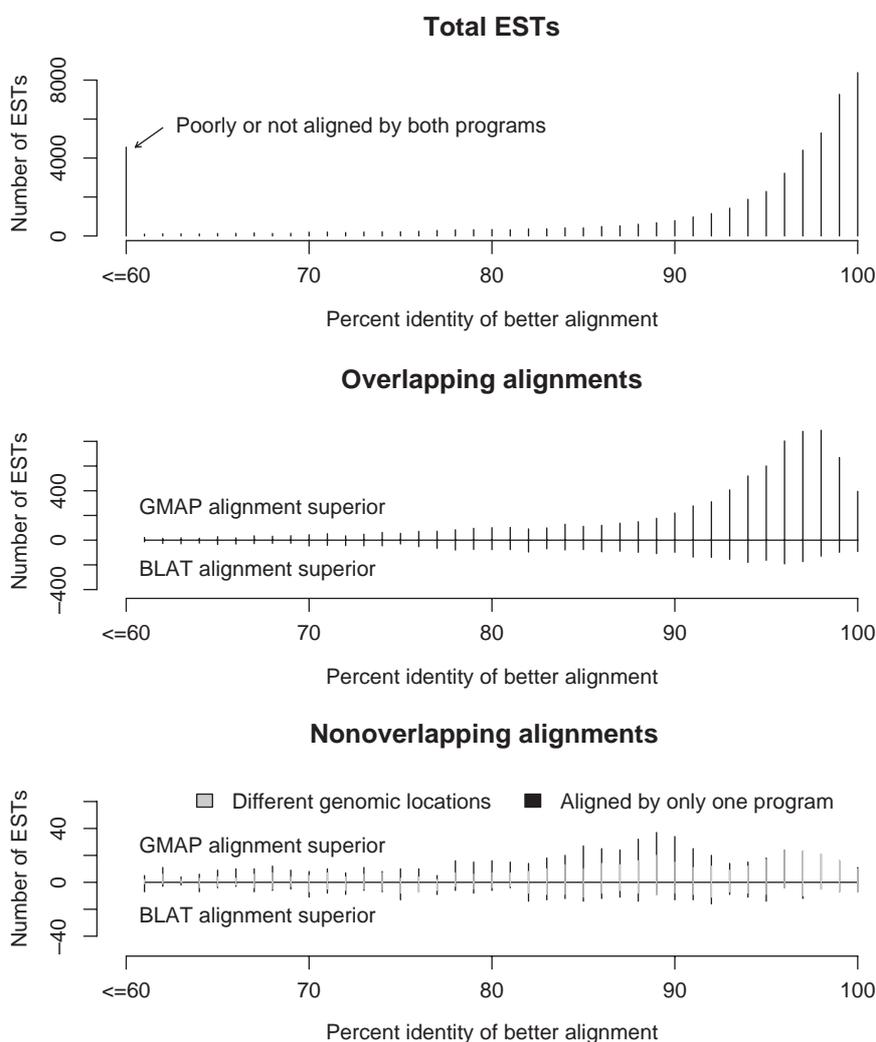
Total ESTs

Overlapping alignments

Nonoverlapping alignments

Fig. 6. Comparison of EST alignment quality between GMAP and BLAT. The top graph shows the total counts of ESTs at various quality levels. The middle graph shows the counts of ESTs whose genomic location by both programs overlap. Counts of ESTs that favor GMAP point upward and those that favor BLAT point downward. The bottom graph shows the distribution for the relatively few non-overlapping ESTs that either have different genomic locations predicted by the two programs, or that are aligned by only one program.

A.thaliana having shorter intron lengths on average than *Saccharomyces cerevisiae* and human beings, and lower organisms only rarely having introns of the 1000-nt or longer variety found commonly in higher eukaryotes (Lim and Burge, 2001). Accordingly, cDNA–genomic alignment programs may potentially perform differently on different species. To assess the performance of GMAP on a species different from the previous human experiments, we evaluated it on the plant *A.thaliana*. We compared GMAP with GeneSeqer (Usuka *et al.*, 2000; Schlueter *et al.*, 2003), which was designed for *Arabidopsis*, and which has been shown in a previous comparison (Haas *et al.*, 2002) to give the best available performance on that genome.

For our test, we used the data set from that comparison, which consisted originally of 5016 full-length cDNAs from Ceres, of which 5000 are publicly available in GenBank and 16 are proprietary; for our purposes, we used only the publicly available sequences. We used GMAP to map and align the cDNAs to the *Arabidopsis* genome (The Arabidopsis Genome Initiative, 2000). We also processed each

cDNA and its corresponding genomic segment, with an additional 1000 nt on each end, using GeneSeqer (5 May 2004 version). We ran GeneSeqer with the flag '-s Arabidopsis' to use its Arabidopsis-specific parameters. (In passing, we note that the parameters for GeneSeqer that we used for the human ESTs in Figures 4 and 5 were '-s human -x 30 -y 60', as recommended by the program's authors.) Running times for the *Arabidopsis* data set were 42 min for GeneSeqer and 1 min for GMAP; these times are not entirely comparable, because GeneSeqer needed to be restarted for each cDNA–genomic alignment.

In all but 23 sequences (or 99.5% of the time), the two programs gave similar gene structures and splice sites. In terms of gene structure, five differences involved short 5′ exons. GeneSeqer reported three 5′ exons not reported by GMAP, with lengths of 9, 6 and 6 nt. GMAP reported two 5′ exons not reported by GeneSeqer, with lengths of 9 and 8 nt. Two of the sequence differences involved 3′ exons. In AY086334, GeneSeqer reports a 9-nt terminal exon not reported by
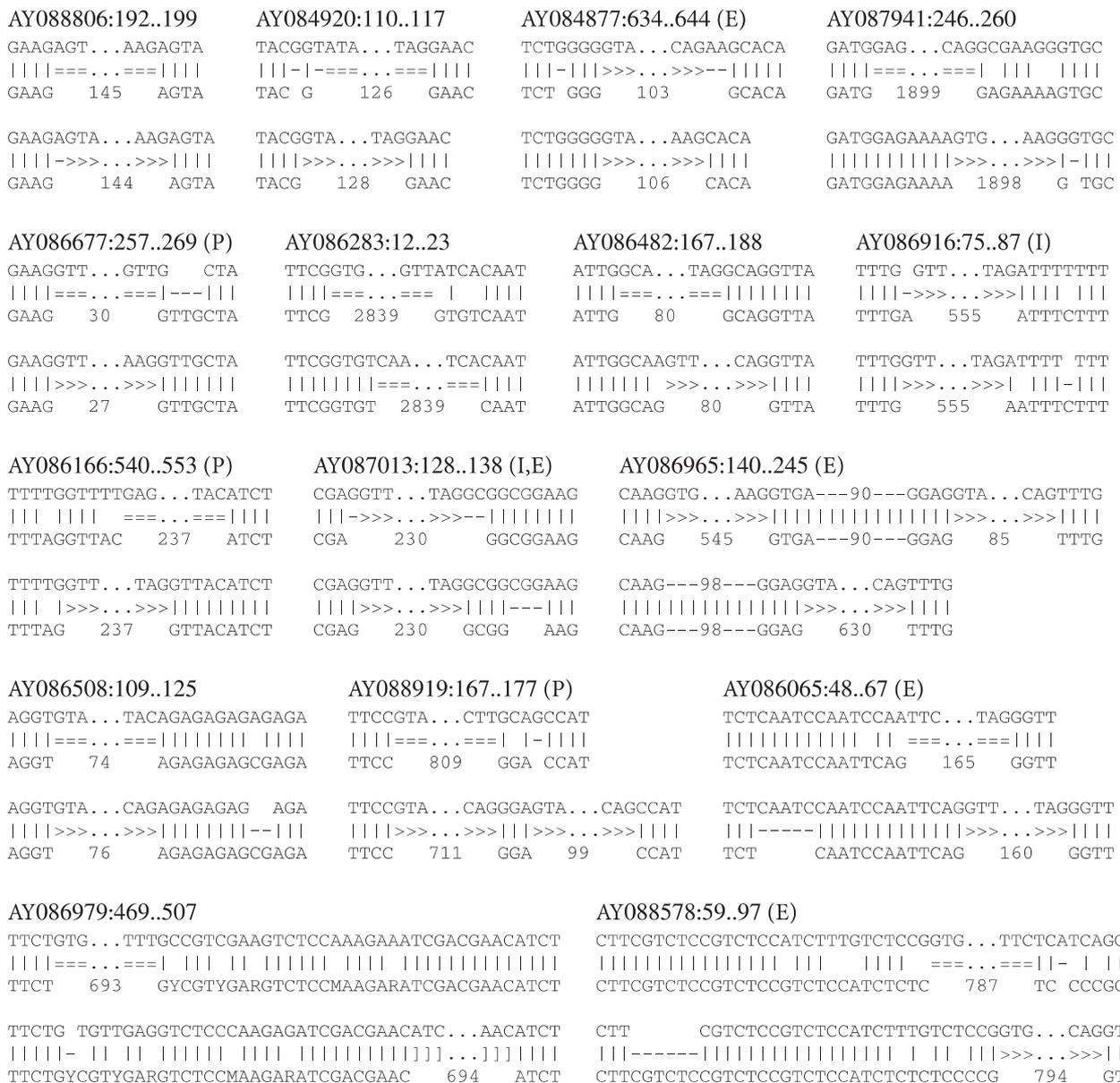
```
AY088806:192..199          AY084920:110..117          AY084877:634..644 (E)          AY087941:246..260

GAAGAGT...AAGAGTA          TACGGTATA...TAGGAAC        TCTGGGGGTA...CAGAAGCACA        GATGGAG...CAGGCGAAGGGTGC
||||===...===||||          |||-|-===...===||||        |||-|||>>>...>>>--|||||        ||||===...===| ||| ||||
GAAG   145   AGTA          TAC G   126   GAAC         TCT GGG   103      GCACA        GATG  1899    GAGAAAGTGC

GAAGAGTA...AAGAGTA         TACGGTA...TAGGAAC          TCTGGGGGTA...AAGCACA          GATGGAGAAAGTG...AAGGGTGC
||||->>>...>>>||||         ||||>>>...>>>||||          |||||||>>>...>>>||||          |||||||||||>>>...>>>|-|||
GAAG   144   AGTA          TACG   128   GAAC           TCTGGG   106   CACA           GATGGAGAAAA   1898   G TGC


AY086677:257..269 (P)      AY086283:12..23            AY086482:167..188              AY086916:75..87 (I)

GAAGGTT...GTTG   CTA        TTCGGTG...GTTATCACAAT      ATTGGCA...TAGGCAGGTTA         TTTG GTT...TAGATTTTTTT
||||===...===|---|||        ||||===...=== | ||||       ||||===...===|||||||||        ||||->>>...>>>|||| |||
GAAG   30   GTTGCTA         TTCG   2839   GTGTCAAT     ATTG   80   GCAGGTTA           TTTGA   555   ATTTCTTT

GAAGGTT...AAGGTTGCTA        TTCGGTGTCAA...TCACAAT      ATTGGCAAGTT...CAGGTTA         TTTGGTT...TAGATTT TTT
||||>>>...>>>|||||||        ||||||||===...===||||      ||||||| >>>...>>>||||          ||||>>>...>>>| |||-|||
GAAG   27   GTTGCTA         TTCGGTGT   2839   CAAT      ATTGGCAG   80   GTTA          TTTG   555   AATTTCTTT


AY086166:540..553 (P)      AY087013:128..138 (I,E)    AY086965:140..245 (E)

TTTTGGTTTTGAG...TACATCT     CGAGGTT...TAGGCGGCGGAAG    CAAGGTG...AAGGTGA---90---GGAGGTA...CAGTTTG
||| |||| ===...===||||      |||->>>...>>>--||||||||    ||||>>>...>>>|||||||||||||||||||>>>...>>>||||
TTTAGGTTAC   237   ATCT     CGA   230   GGCGGAAG       CAAG   545   GTGA---90---GGAG   85   TTTG

TTTTGGTT...TAGGTTACATCT     CGAGGTT...TAGGCGGCGGAAG    CAAG---98---GGAGGTA...CAGTTTG
||| |>>>...>>>||||||||      ||||>>>...>>>||||---|||     |||||||||||||||||>>>...>>>||||
TTTAG   237   GTTACATCT     CGAG   230   GCGG   AAG    CAAG---98---GGAG   630   TTTG


AY086508:109..125          AY088919:167..177 (P)      AY086065:48..67 (E)

AGGTGTA...TACAGAGAGAGAGAGA  TTCCGTA...CTTGCAGCCAT      TCTCAATCCAATCCAATTC...TAGGGTT
||||===...===||||||||| |||| ||||===...===| |-||||      |||||||||||| || ===...===||||
AGGT   74   AGAGAGAGCGAGA   TTCC   809   GGA CCAT       TCTCAATCCAATTCAG   165   GGTT

AGGTGTA...CAGAGAGAGAG  AGA  TTCCGTA...CAGGGAGTA...CAGCCAT  TCTCAATCCAATCCAATTCAGGTT...TAGGGTT
||||>>>...>>>|||||||||--|||  ||||>>>...>>>|||>>>...>>>||||  |||-----|||||||||||||||>>>...>>>||||
AGGT   76   AGAGAGAGCGAGA   TTCC   711   GGA   99   CCAT    TCT   CAATCCAATTCAG   160   GGTT


AY086979:469..507                                     AY088578:59..97 (E)

TTCTGTG...TTTGCCGTCGAAGTCTCCAAAGAAATCGACGAACATCT      CTTCGTCTCCGTCTCCATCTTTGTCTCCGGTG...TTCTCATCAGGTAA
||||===...===| ||| || ||||| |||| ||||||||||||||      ||||||||||||||| ||| |||| ===...===||- | |||||
TTCT   693   GYCGTYGARGTCTCCMAAGARATCGACGAACATCT       CTTCGTCTCCGTCTCCGTCTCCATCTCTC   787   TC CCCGGTAA

TTCTG TGTTGAGGTCTCCCAAGAGATCGACGAACATC...AACATCT      CTT     CGTCTCCGTCTCCATCTTTGTCTCCGGTG...CAGGTAA
|||||- || || |||||| |||| ||||||||||||]]]...]]]||||    |||------|||||||||||||||||| | || |||>>>...>>>||||
TTCTGYCGTYGARGTCTCCMAAGARATCGACGAAC   694   ATCT      CTTCGTCTCCGTCTCCGTCTCCATCTCTCTCCCCG   794   GTAA
```

Fig. 7. Comparison between GeneSeqer and GMAP. These alignments represent all splicing differences between the two programs on a dataset of 5000 *Arabidopsis* cDNAs. Each comparison shows the GeneSeqer alignment on top and the GMAP alignment on bottom. Notation is the same as in Figure 4, with the addition of the ']' character to indicate an AT–AC intron, and a compressed view of exons for AY086965. Footnotes after the accession numbers indicate whether: (I) genomic splice site predictions are identical and difference exists only in the exon–exon boundary, (P) same answers are given when parameters are changed in GeneSeqer, (E) EST or mRNA evidence is available for the splice site.

GMAP. In AY085991, GMAP found a 33-nt terminal exon not found by GeneSeqer. Instead, GeneSeqer extends the previous exon through a stretch of 2 gaps and 14 mismatches.

The remaining 16 cases involve differences in splice sites and one microexon. The alignments for these cases are shown in Figure 7. In two of these cases, AY086916 and AY87013, marked in Figure 7 with an (I), the genomic splice site predictions of the two programs are identical, and the differences lie only in the predicted exon–exon boundary.

In three cases, marked with a (P), a different choice of parameters allows GeneSeqer to give the same answer as GMAP. For AY08166,

GeneSeqer makes an alignment on the wrong strand. Strand selection in GeneSeqer depends on splice site scores, and the correct strand gives very poor splice sites. For AY086677, the intron shown is shorter than the minimum length that is the default in GeneSeqer; as discussed previously, such short introns are atypical. And for AY088919, the 3-nt microexon is shorter than the default minimum size of 5 in GeneSeqer.

In evaluating the remaining differences, we should note that the ecotypes of the data set sequences do not necessarily correspond to the Columbia ecotype used in assembling the genome. Therefore, mismatches and gaps may reflect differences in ecotype. GeneSeqer

is more likely to introduce substitutions and gaps around introns, because it depends upon probabilistic splice site models in addition to the sequence data, whereas GMAP tries to identify the most parsimonious alignment of the given cDNA to the given genomic segment.

To help determine which program gives the correct result, we looked for supporting evidence from other ESTs or mRNAs that map to the splice site. Supporting evidence was found for five cases, marked in Figure 7 with an (E). For AY086065 and AY088578, the evidence appears to support the splice site in the GMAP alignment, whereas for AY084877 and AY087013, the evidence appears to support the GeneSeqer splice site. For AY086965, the EST and mRNA evidence do not resolve the issue of where the cDNA nucleotides map to the genome. In this case, we found other sequences with an additional 462 nucleotides relative to the test cDNA, which GMAP introduces as a new middle exon and which GeneSeqer appends to its existing middle exon. We also found a full-length sequence AAC50956 in the patent database that GeneSeqer aligns to give the same single 630-nt intron as GMAP.

To evaluate the robustness of the two programs to sequence error, we created mutated data sets at rates of 1 and 3%, using the same approach as in Experiment 1. We used only the 4977 sequences for which the two programs agreed on gene structure. The results of the two programs were roughly equivalent. On the 1% data set, GeneSeqer had no gene structure errors and shifted canonical splice sites in 8 sequences. In comparison, GMAP had 4 gene structure errors and 3 sequences with splicing errors. The gene structure errors committed by GMAP were relatively minor, with missing 5' exons of lengths 10 and 9 nt and missing internal exons of 6 and 7 nt.

On the 3% data set, GeneSeqer also had no gene structure errors and shifted canonical splice sites in 13 sequences. GMAP had gene structure errors in 9 sequences, and splicing errors in 9 sequences. As before, the gene structure errors by GMAP were minor with missing 5' exons of lengths 20, 10 and 9 nt, and 6 missing internal exons, all of length 7 nt or less. The splicing errors by GMAP involved shifted canonical splice sites in 5 sequences, and conversion of semi-canonical (GC–AG) splice sites to canonical ones in 4 sequences.

IMPLEMENTATION

GMAP is implemented in the C programming language. It can be compiled and run on any modern Unix system with a 32-bit or higher architecture. We have compiled and run the code successfully on Digital Alpha Tru64, Silicon Graphics Irix, Sun Solaris, Intel Linux and Mac OS X platforms. Source code and documentation for GMAP and associated programs are available for open use at http://www.gene.com/share/gmap

Before GMAP can handle a given genome, it requires that the genome be pre-processed, by constructing a genomic oligomer index (consisting of an offset file and a position file) and a genomic sequence file. These files are generated by an auxiliary program GMAP_SETUP, in a process that can require a few hours to complete, depending on the size of the genome. However, each release of a genome needs to be set up only once, and the resulting binary files are portable across different computer architectures, because GMAP translates the file contents as necessary for big-endian and little-endian platforms.

The genome may be read in as FASTA files that contain either contigs in any order or entire assembled chromosomes. The chromosomal location of contigs can be specified either in the header lines of the FASTA file or in a separate file. The program can handle an arbitrary number of chromosomes, and can concatenate a collection of contigs to create a special-purpose chromosome (e.g., '22U' for unmapped contigs from chromosome 22). Arbitrary chromosomes may also be used to include alternate versions of a chromosome, such as the Celera version of mouse chromosome 16 (Mural *et al.*, 2002) or the two available versions of human chromosome 7 (Scherer *et al.*, 2003; Hillier *et al.*, 2003).

The genomic sequence file represents the genome in one continuous sequence with the chromosomes concatenated. It may be stored in a compressed format, which facilitates the reading of the entire genome into RAM, when sufficient RAM is available. (In addition, some 32-bit machines limit file offsets to 2 GB, which makes compression necessary on these machines for random file access functions to work properly.) The compressed format allocates 3 bits per position, allowing for representation of A, C, G, T, N and X. The compression scheme stores each block of 32 nucleotides into three 32-bit words, with the first two words holding the first two bits of each nucleotide, and the last word holding the third bit (which is set only for non-ACGT letters). If the user chooses not to compress the genomic sequence file, the full range of alphabetic characters can of course be represented. For the human genome, genomic oligomer files require a total of 1.9 GB and the genomic sequence file requires 1.1 GB compressed (3.1 GB uncompressed). Once a genome is processed by GMAP_SETUP, the user may retrieve arbitrary segments from the genome using the auxiliary program GET-GENOME. The segments may be specified by either contig coordinates (if applicable) or chromosomal coordinates.

GMAP can be run on a FASTA file containing one or more cDNA sequences. For a single sequence, GMAP is generally run in interactive mode, in which parts of the genomic files are read directly as needed. For larger runs, the user may select a batch mode, in which the program attempts memory mapping, first on the oligomer file and then on the sequence file. Memory mapping permits fast access to portions of a file, without having to load the entire file or allocate dedicated RAM for the entire file. If the attempt at memory mapping fails on either file (usually due to insufficient memory available), the program automatically resorts to its interactive mode for that file, by subsequently using file access functions. Memory mapping of the position file, which needs more frequent access, is more important for speed than memory mapping of the genome file. Therefore, the minimum memory requirement for the program is only 128 MB, although the batch mode works optimally when there is enough memory available (2 GB or more) to permit memory mapping of the position file and to avoid having to swap out parts of that file.

A third mode of GMAP allows the user to provide both a genomic segment and one or more cDNA sequences. In this mode, oligomer index files are not needed, because GMAP bypasses the mapping step and aligns the cDNA sequences to the given segment. This mode gives GMAP the same functionality as pure genomic alignment programs, and is useful for computing cDNA alignments on the fly for a particular genomic region of interest.

In processing a FASTA file, GMAP is able to use multithreading, which works most effectively on machines with multiple processors. When multithreading is enabled, one thread handles reading of the input, one handles writing of the output alignments, and one or more worker threads each processes an individual cDNA sequence. Our implementation allocates thread-specific memory

in critical portions to reduce memory contention across threads. Thread-specific memory allocation is particularly critical because the underlying representation of a cDNA–genomic alignment in GMAP is a linked list, which requires cells to be added and deleted frequently. Although this representation facilitates the insertion, deletion and substitution of subalignments, it can cause contention for heap memory when multiple threads need to build up their linked lists simultaneously. In turn, heap contention prevents multithreading from using the full potential of multiple parallel processors. Therefore, GMAP has dedicated memory allocation procedures that give each thread its own pool of heap memory as needed, thereby minimizing heap contention.

Additional features

In the course of our development and use of GMAP, we have added several features that extend its functionality. Although a full discussion of these features is beyond the scope of this paper, we mention them briefly here.

Identification of chimeric ESTs   GMAP is capable of finding and reporting chimeras, or ESTs whose 5′ and 3′ ends map to different genomic regions. Although chimeric ESTs are often thought to be library artifacts (Sorek and Safer, 2003), some observations suggest that such chimeras are indicative of translocation events in cancer (Panagopoulos *et al.*, 2000). Furthermore, some recent experimental results have confirmed some novel chimeras detected using genomic alignments of ESTs (Hahn *et al.*, 2004).

For a given cDNA, GMAP maintains all alignment results in memory during its calculations. When no single alignment is able to cover a certain fraction of the original length (specified by the user, with a reasonable value being 60 or 70%), GMAP finds the pair of partial alignments that provides the greatest coverage of the query sequence. This pair is then reported as the optimal solution to the chimeric alignment. In some cases, this solution has better coverage than one would get by taking the longest alignment and then trying to align the remaining cDNA.

Relative alignment of ESTs   GMAP has a mode where a set of ESTs can be aligned relative to a reference sequence. In this mode, the user provides GMAP with both a full-length mRNA and a FASTA file of ESTs. GMAP then uses the mRNA to identify the appropriate genomic segment and to mark it with the coding region and codon positions. Finally, GMAP uses this marked genomic segment to align the ESTs. Based on the codon markings, GMAP can determine whether each EST overlaps the coding region or lies in an untranslated region or an intron.

The genomic codon boundaries also enable GMAP to perform a frameshift-tolerant translation of the EST. This translation maximizes the amount of EST information available to identify putative point mutations and polymorphisms, but of course misses potentially true frameshift mutations that may lead to a premature stop codon. GMAP compares the translation of each EST against the translation of the reference sequence to report a summary of protein sequence variations, including SNPs, amino acid insertions and deletions, and alternative splice forms.

Compressed alignment format   GMAP can produce alignments in a variety of formats, including a compressed format that saves considerable space. The compressed format stores only differences relative to the genomic sequence. The compressed alignments may be uncompressed to their original form using a provided utility program.

Lookup of genomic map information   GMAP has the capability of looking up information in a genomic map file to find information relative to a given cDNA alignment. Genomic map files consist of a set of genomic intervals, with each interval having some annotation, an optional label and an optional tag. They are implemented as an integer interval tree (IIT) (Bucher and Edelsbrunner, 1983), which is a binary tree structure designed for the rapid retrieval of all intervals that overlap a given query interval. IITs permit retrieval of all k overlapping intervals for a given query interval in $O(k + \log_2 n)$, where n is the total number of intervals in the database.

The annotations in our genomic map files can be of arbitrary length, meaning that one may store sequences, entire alignments, or other arbitrary genomic bounds such as cytogenic bands and syntenic regions. At our institution, we routinely create genomic map files containing previously computed EST alignments (in our compressed format). Such files allow us to rapidly retrieve all ESTs that overlap a given mRNA on the genome. Another use is the construction of a genomic map file with gene boundaries (potentially overlapping). The resulting map file can then be used to tell which gene a given EST belongs to.

Each interval in our genomic map files may have a label, and the set of labels are also stored in a binary tree structure, allowing one to retrieve an interval by name in logarithmic time. Tags allow one to mark and retrieve subsets of intervals. We often use tags to store intervals as being on the plus or minus strand of the genome, so we can retrieve ESTs from a specified strand if desired.

Aligning against multiple strains   Our IIT files also make it possible to efficiently store and retrieve strain variants for a given species. Therefore, we have built into GMAP the ability to map and align a given cDNA over multiple strains simultaneously. Mapping over multiple strains requires that we augment our genomic index table with 24-mers from all strains. Alignment over multiple strains requires us to build a genomic map file that contains strain differences and their genomic coordinates on a reference strain. At run time, when a candidate genomic segment is found in the mapping step, GMAP uses in subsequent alignment steps not only the genomic segment from the reference strain but also segments from relevant alternate strains by patching in the alternate strain sequence. In ranking the results, GMAP is therefore able to identify the strain that best matches a given cDNA. Given that the NCBI mouse genome (build 33) has sequences from 9 different strains, this feature can result in considerable savings over mapping and aligning repeatedly against a complete genome for each mouse strain.

DISCUSSION

Our program GMAP is designed to provide a general-purpose solution for cDNA–genomic mapping and alignment. Most existing programs are intended to solve either the mapping task or the alignment task, but not both. Programs that do provide integrated mapping and alignment, namely, BLAT and SQUALL, are intended primarily for batch or server mode, not for single query or interactive use.

Although one advantage of an integrated mapping and alignment program over separate programs is convenience, coupling of the mapping and alignment tasks also provides functional advantages. Genomic alignment programs require the user to supply the correct

genomic segment to align to, but the correct segment may not be apparent when there are multiple candidate genomic locations. One approach to this problem is to try to improve the ability of the genomic mapping procedure to find the correct location initially. Another approach, taken by GMAP, exploits the integration of genomic mapping and alignment: the correct genomic mapping is determined by the results of the alignment procedure.

Our program GMAP has been in development for over 3 years. In that time, the program has undergone continual evolution, both to improve its accuracy and speed, and to provide additional functionality. Since existing cDNA–genomic alignment programs can handle 95% of error-free sequences correctly, our work has been devoted primarily to the remaining 5%, and toward perfecting gene structure determination and splice site detection in the presence of sequence error.

One central issue in our development process has been whether to use probabilistic models of splice sites, such as scoring matrices (Salzberg, 1997; Brendel and Kleffe, 1998). Such models are used widely in *ab initio* gene finding programs (Uberbacher and Mural, 1991; Burge and Karlin, 1997; Lukashin and Borodovsky, 1998; Salzberg *et al.*, 1999; Reese *et al.*, 2000) and in homology-based gene finding programs (Guigó *et al.*, 1992; Huang *et al.*, 1997; Gotoh, 1999; Batzoglou *et al.*, 2000; Korf *et al.*, 2001; Novichkov *et al.*, 2001; Rinner and Morgenstern, 2002; Brendel *et al.*, 2004), and their use has continued in many cDNA–genomic alignment programs. Likewise, in our early development of GMAP, we also used such scoring matrices. However, when the alignment methodology was improved through oligomer chaining and sandwich DP, such matrices proved to be unnecessary, since the cDNA sequence (even with errors) plus the dinucleotide pairs at the end of the introns provide enough information to determine the splice site boundaries accurately. The absence of splice site models provides some advantages: it allows the reported alignment to reflect the given data rather than prior probabilities of splice site patterns, and it potentially makes the genomic alignment task generalizable across species.

Another issue in our development work has been computational speed, both for processing a single cDNA and for processing a large batch of ESTs. We have found it convenient to be able to map and align a single cDNA sequence quickly when needed, and to be able to switch quickly among different genomes or versions of a genome. As the number of sequenced genomes grows, a file-based approach to mapping and alignment should become increasingly useful. Although we have mentioned batch running times only briefly in our experimental results, they show that GMAP provides a several-fold increase in speed over existing programs. Our running times are even faster when multithreading is enabled. In our institution, we are able to map and align the GenBank set of approximately 6 million human ESTs onto the genome using a single computer with three worker threads in less than 2 days.

Although some users may not be concerned directly with computational issues such as alignment accuracy or speed, these issues can have a significant impact on our understanding of the underlying biology. As we have mentioned, improvements in alignment accuracy can lead to improved genomic mapping of cDNAs, and hence result in better definitions of gene boundaries. Moreover, the ability of an alignment program to detect splice sites in the presence of sequence error and in the absence of prior bias may alter our assessment of the frequencies of canonical, semi-canonical and non-canonical splice sites. Likewise, our knowledge of microexons and chromosomal rearrangements can be enhanced by accurate prediction of gene structures and chimeric ESTs. Accurate and fast genomic mapping and alignment should facilitate our exploration of the genome and our understanding of the structure, function and evolution of genes.

REFERENCES

Allen,J.E. *et al.* (2004) Computational gene prediction using multiple sources of evidence. *Genome Res.*, **14**, 142–148.

Altschul,S.F. *et al.* (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Atteson,K. (1998) Calculating the exact probability of language-like patterns in biomolecular sequences. In *Proceedings of the Sixth International Conference on Intelligent Systems in Molecular Biology*, AAAI Press, Menlo Park, CA, pp. 17–24.

Bailey,J.A. *et al.* (2002) Recent segmental duplications in the human genome. *Science*, **297**, 1003–1007.

Batzoglou,S. *et al.* (2000) Human and mouse gene structure: Comparative analysis and application to exon prediction. *Genome Res.*, **10**, 950–958.

Birney,E. *et al.* (2004) GeneWise and Genomewise. *Genome Res.*, **14**, 988–995.

Boyer,R.S. and Moore,J.S. (1977) A fast string searching algorithm. *Commun. ACM*, **20**, 762–772.

Bray,N. *et al.* (2003) AVID: a global alignment program. *Genome Res.*, **13**, 97–102.

Brendel,V. and Kleffe,J. (1998) Prediction of locally optimal splice sites in plant pre-mRNA with applications to gene identification in *Arabidopsis thaliana* genomic DNA. *Nucleic Acids Res.*, **26**, 4748–4757.

Brendel,V. *et al.* (2004) Gene structure prediction from consensus spliced alignment of multiple ESTs from matching the same genomic locus. *Bioinformatics*, **20**, 115–1169.

Brudno,M. *et al.* (2003a) Fast and sensitive multiple alignment of large genomic sequences. *BMC Bioinformatics*, **4**, 66.

Brudno,M. *et al.* (2003b) LAGAN and Multi-LAGAN: Efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res.*, **13**, 721–731.

Bucher,W. and Edelsbrunner,H. (1983) On expected- and worst-case segment trees. In Preparata,F.P. (ed.), *Advances in Computing Research*, Vol. 1. Jai Press, London, pp. 109–125.

Burge,C. and Karlin,S. (1997) Prediction of complete gene structures in human genomic DNA. *J. Mol. Biol.*, **268**, 78–94.

Burset,M. and Guigó, R. (1996) Evaluation of gene structure prediction programs. *Genomics*, **34**, 353–357.

Burset,M. *et al.* (2000) Analysis of canonical and non-canonical splice sites in mammalian genomes. *Nucleic Acids Res.*, **28**, 4364–4375.

Burset,M. *et al.* (2001) SpliceDB: database of canonical and non-canonical mammalian splice sites. *Nucleic Acids Res.*, **29**, 255–259.

Carlo,T. *et al.* (2000) A 5′ splice site-proximal enhancer binds SF1 and activates exon bridging of a microexon. *Mol. Cell. Biol.*, **20**, 3988–3995.

Chain,P. *et al.* (2003) An applications-focused review of comparative genomics tools: Capabilities, limitations and future challenges. *Brief. Bioinformatics*, **4**, 105–123.

Clamp,M. *et al.* (2003) Ensembl 2002: accommodating comparative genomics. *Nucleic Acids Res.*, **31**, 38–42.

Delcher,A.L. *et al.* (1999) Alignment of whole genomes. *Nucleic Acids Res.*, **27**, 2369–2376.

Delcher,A.L. *et al.* (2002) Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Res.*, **30**, 2478–2483.

Drabenstot,S.D. *et al.* (2003) FELINES: a utility for extracting and examining EST-defined introns and exons. *Nucleic Acids Res.*, **31**, e141.

Florea,L. *et al.* (1998) A computer program for aligning a cDNA sequence with a genomic DNA sequence. *Genome Res.*, **8**, 967–974.

Florea,L. *et al.* (2005) Gene and alternative splicing annotation with AIR. *Genome Res.*, **15**, 54–66.

Gelfand,M.S. (1989) Statistical analysis of mammalian pre-mRNA splicing sites. *Nucleic Acids Res.*, **17**, 6369–6382.

Gelfand,M.S. *et al*. (1996) Gene recognition via spliced alignment. *Proc. Natl Acad. Sci. USA*, **93**, 9061–9066.

Gotoh,O. (1999) Homology-based gene structure prediction: simplified matching algorithm using a translated codon (tron) and improved accuracy by allowing for long gaps. *Bioinformatics*, **16**, 190–202.

Guigó,R. *et al*. (1992) Prediction of gene structure. *J. Mol. Biol.*, **226**, 141–157.

Haas,B.J. *et al*. (2002) Full-length messenger RNA sequences greatly improve genome annotation. *Genome Biol.*, **3**, research0029.1–0029.12.

Haas,B.J. *et al*. (2003) Improving the *Arabidopsis* genome annotation using maximal transcript alignment assemblies. *Nucleic Acids Res.*, **31**, 5654–5666.

Hahn,Y. *et al*. (2004) Finding fusion genes resulting from chromosome rearrangement by analyzing the expressed sequence databases. *Proc. Natl Acad. Sci. USA*, **101**, 13257–13261.

Hillier,L.W. *et al*. (2003) The DNA sequence of human chromosome 7. *Nature*, **424**, 157–164.

Huang,X. (1996) Fast comparison of a DNA sequence with a protein sequence database. *Microb. Comp. Genomics*, **1**, 281–291.

Huang,X. *et al*. (1997) A tool for analyzing and annotating genomic sequences. *Genomics*, **46**, 37–45.

Irizarry,K. *et al*. (2000) Genome-wide analysis of single-nucleotide polymorphisms in human expressed sequences. *Nature Genet.*, **26**, 233–236.

Jiang,J. and Jacob,H.J. (1998) EbEST: an automated tool using expressed sequence tags to delineate gene structure. *Genome Res.*, **8**, 268–275.

Kalafus,K.J. *et al*. (2004) Pash: efficient genome-scale anchoring by positional hashing. *Genome Res.*, **14**, 672–678.

Kan,Z. *et al*. (2001) Gene structure prediction and alternative splicing analysis using genomically aligned ESTs. *Genome Res.*, **11**, 889–900.

Kan,Z. *et al*. (2002) Selecting for functional alternative splices in ESTs. *Genome Res.*, **12**, 1837–1845.

Kent,W.J. (2002) BLAT—the BLAST-like alignment tool. *Genome Res.*, **12**, 656–664.

Kent,W.J. and Zahler,A.M. (2000) Conservation, regulation, synteny, and introns in large-scale *C.briggsae—C.elegans* genomic alignment. *Genome Res.*, **10**, 1115–1125.

Kim,N. *et al*. (2004) ASmodeler: gene modeling of alternative splicing from genomic alignment of mRNA, EST and protein sequences. *Nucleic Acids Res.*, **32**, W181–W186.

Korf,I. *et al*. (2001) Integrating genomic homology into gene structure prediction. *Bioinformatics*, **17**, S140–S148.

Lee,B.T.K. *et al*. (2003) MGAlignIt: a web service for the alignment of mRNA/EST and genomic sequences. *Nucleic Acids Res.*, **31**, 3533–3536.

Lim,L.P. and Burge,C.B. (2001) A computational analysis of sequence features involved in recognition of short introns. *Proc. Natl Acad. Sci. USA*, **98**, 11193–11198.

Lukashin,A.V. and Borodovsky,M. (1998) GeneMark.HMM: new solutions for gene finding. *Nucleic Acids Res.*, **26**, 1107–1115.

Ma,B. *et al*. (2002) PatternHunter: faster and more sensitive homology search. *Bioinformatics*, **18**, 440–445.

Manber,U. and Myers,G. (1993) Suffix arrays: a new method for on-line string searches. *SIAM J. Comput.*, **22**, 935–948.

McAllister,L. *et al*. (1992) Alternative splicing of micro-exons creates multiple forms of the insect cell adhesion molecular fasciclin I. *J. Neurosci.*, **12**, 895–905.

Milanesi,L. and Rogozin,I.B. (2003) ESTMAP: a system for expressed sequence tags mapping on genomic sequences. *IEEE Trans. Nanobiosci.*, **2**, 75–78.

Modrek,B. and Lee,C. (2002) A genomic view of alternative splicing. *Nature Genet.*, **30**, 13–19.

Morgenstern,B. (1999) DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, **15**, 211–218.

Mott,R. (1997) EST_GENOME: a program to align spliced DNA sequences to unspliced genomic DNA. *Comput. Appl. Biosci.*, **13**, 477–478.

Mount,S.M. (1981) A catalogue of splice junction sequences. *Nucleic Acids Res.*, **10**, 459–472.

Mural,R.J. *et al*. (2002) A comparison of whole-genome shotgun-derived mouse chromosome 16 and the human genome. *Science*, **296**, 1661.

Nakata,K. *et al*. (1985) Prediction of splice junctions in mRNA sequences. *Nucleic Acids Res.*, **13**, 5327–5340.

Needleman,S.B. and Wunsch,C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, **48**, 443–453.

Ning,Z. *et al*. (2001) SSAHA: a fast search method for large DNA databasees. *Genome Res.*, **11**, 1725–1729.

Novichkov,P.S. *et al*. (2001) Gene recognition in eukaryotic DNA by comparison of genomic sequences. *Bioinformatics*, **17**, 1011–1018.

Ogasawara,J. and Morishita,S. (2002) Fast and sensitive algorithm for aligning ESTs to human genome. In *Proceedings of the First IEEE Computer Society Bioinformatics Conference*, Stanford, CA, pp. 43–53.

Panagopoulos,I. *et al*. (2000) A novel FUS/CHOP chimera in myxoid liposarcoma. *Biochem. Biophys. Res. Commun.*, **279**, 838–845.

Ranganathan,S. *et al*. (2003) MGAlign, a reduced search space approach to the alignment of mRNA sequences to genomic sequences. In *Proceedings of the 14th International Conference on Genome Informatics*, Yokohama, Japan, Universal Academy Press, Tokyo, Vol. 14, pp. 474–475.

Reese,M.G. *et al*. (2000) Genie—gene finding in *Drosophila melanogaster*. *Genome Res.*, **10**, 529–538.

Richterich,P. (1998) Estimation of errors in 'raw' DNA sequences: a validation study. *Genome Res.*, **8**, 251–259.

Rinner,O. and Morgenstern,B. (2002) AGenDA: gene prediction by comparative sequence analysis. *In Silico Biol.*, **2**, 195–205.

Salzberg,S.L. (1997) A method for identifying splicing sites and translation start sites in eukaryotic mRNA. *Comput. Appl. Biosci.*, **13**, 365–376.

Salzberg,S.L. *et al*. (1999) Interpolated Markov models for eukaryotic gene finding. *Genomics*, **59**, 24–31.

Sankoff,D. and Kruskal,J.B. (eds) (1999) *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison.* CSLI Publications, Stanford, CA.

Scherer,S.W. *et al*. (2003) Human chromosome 7: DNA sequence and biology. *Science*, **300**, 767–772.

Schlueter,S.D. *et al*. (2003) GeneSeqer@PlantDB: gene structure prediction in plant genomes. *Nucleic Acids Res.*, **31**, 3597–3600.

Schwartz,S. *et al*. (2000) PipMaker—a web server for aligning two genomic DNA sequences. *Genome Res.*, **10**, 577–586.

Senapathy,P. *et al*. (1990) Splice junctions, branch point sites, and exons: sequence statistics, identification, and applications to genome project. *Methods Enzymol.*, **183**, 252–278.

Simpson,C.G. *et al*. (2000) Requirements for mini-exon inclusion in potato invertase mRNAs provides evidence for exon-scanning interactions in plants. *RNA*, **6**, 422–433.

Smith,T.F. and Waterman,M.S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.

Solovyev,V. (2002) Structure, properties and computer identification of eukaryotic genes. In Lengauer,T. (ed.), *Bioinformatics—From Genomes to Drugs*, Vol. 14, *Methods and Principles in Medicinal Chemistry*, Ch. 3. Wiley-VCH, Weinheim, Germany, pp. 59–111.

Sorek,R. and Safer,H.M. (2003) A novel algorithm for computational identification of contaminated EST libraries. *Nucleic Acids Res.*, **31**, 1067–1074.

Sterner,D.A. and Berget,S.M. (1993) In vivo recognition of a vertebrate mini-exon as an exon-intron-exon unit. *Mol. Cell. Biol.*, **13**, 2677–2687.

Tammi,M.T. *et al*. (2003) Correcting errors in shotgun sequences. *Nucleic Acids Res.*, **15**, 4663–4672.

The Arabidopsis Genome Initiative (2000) Analysis of the genome sequence of the flowering plant *Arabidopsis thaliana*. *Nature*, **408**, 796–815.

Uberbacher,E.C. and Mural,R.J. (1991) Locating protein coding regions in human DNA sequences using a multiple sensor-neural network approach. *Proc. Natl Acad. Sci. USA*, **88**, 11261–11265.

Usuka,J. *et al*. (2000) Optimal spliced alignment of homologous cDNA to a genomic DNA template. *Bioinformatics*, **16**, 203–211.

Volfovsky,N. *et al*. (2003) Computational discovery of internal micro-exons. *Genome Res.*, **13**, 1216–1221.

Wheelan,S.J. *et al*. (2001) Spidey: a tool for mRNA-to-genomic alignments. *Genome Res.*, **11**, 1952–1957.

Wheeler,D.L. *et al*. (2003) Database resources of the National Center for Biotechnology. *Nucleic Acids Res.*, **31**, 28–33.

Wiehe,T. *et al*. (2001) SGP-1: prediction and validation of homologous genes based on sequence alignments. *Genome Res.*, **11**, 1574–1583.

Wieringa,B. *et al*. (1984) A minimal intron length but no specific internal sequence is required for splicing the large rabbit beta-globin intron. *Cell*, **37**, 915–925.

Yeo,G. *et al*. (2004) Variation in alternative splicing across human tissues. *Genome Biol.*, **5**, R74.

Yu,J. *et al*. (2002) Minimal introns are not 'junk'. *Genome Res.*, **12**, 1185–1189.

Zavolan,M. *et al*. (2002) Splice variation in mouse full-length cDNAs identified by mapping to the mouse genome. *Genome Res.*, **12**, 1377–1385.

Zhang,H. (2003) Alignment of BLAST high-scoring alignment pairs based on the longest increasing subsequence algorithm. *Bioinformatics*, **19**, 1391–1396.

Zhang,Z. and Gerstein,M. (2004) Large-scale analysis of pseudogenes in the human genome. *Curr. Opin. Genet. Develop.*, **14**, 328–335.

Zhang,Z. *et al*. (2000) A greedy algorithm for aligning DNA sequences. *J. Comput. Biol.*, **7**, 203–214.

Zhuo,D. *et al*. (2003) Assembly, annotation, and integration of UNIGENE clusters into the human genome draft. *Genome Res.*, **11**, 904–918.