The Biologist's Guide to Paracel's Similarity Search Algorithms

Introduction

Many biological questions require the comparison of one or more sequences to each other. The nature of those comparisons depends on the question being asked, the time allowed to answer the question, the manner in which the answers will be used in subsequent analyses, the required accuracy of the answer, and so on.

Fundamentally, the purpose of all similarity searches is to measure the "distance" between sequences. However, the meaning of "distance" changes depending on the investigation of interest. For example, a question in which protein hydrophobicity is the basis for comparison will use different metrics and a different algorithm than one in which the presence or absence of a specific binding domain is in question. Understanding when and why a certain algorithm is needed is essential to properly producing the scientific evidence needed for an investigation.

Algorithm selection also requires considering time and accuracy of the result. In some situations a fast but possibly less precise result is more important than a very precise answer that takes far longer. Algorithm precision is measured by two parameters: sensitivity and specificity. Sensitivity is the percentage of true positives found, i.e., the number of correctly identified matches relative to the total number of true matches. Specificity is the number of true matches found relative to the total number of matches reported. Sensitivity and specificity often conflict with each other because higher sensitivity also means that more unrelated sequences are reported.

Lastly, investigations often require independent confirmation from multiple computational or wet lab experiments. As a consequence, algorithm selection may be influenced by the availability and quality of confirming data. A typical example is to compare available EST, cDNA, or protein data with the results of gene prediction studies to confirm the reasonableness of those studies.

What is similarity comparison?

Similarity comparisons evaluate the "closeness" of sequences to each other by computing a metric that reflects a reward for "allowed" differences and a penalty for "disallowed" differences. An "objective function" determines what rewards and penalties are important and how to combine these into the closeness metric.

Consider an example in which a message is received from a spacecraft parked in the red zone around a distant planet. The message traveled millions of miles and was likely

corrupted in the transmission. Our job at the receiver is to determine which message was most likely sent. Suppose that the spacecraft sends only one of four messages:

1. WEATHER IS BEAUTIFUL, WISH YOU WERE HERE
2. WEATHER IS HERE, WISH YOU WERE BEAUTIFUL
3. WISH THERE WAS BEAUTIFUL WEATHER HERE
4. ENTERING SAFEMODE NEED HELP

At each position in the received message, a character may be correct, deleted relative to the original message, or a substitution for another character. We might also know that the chance that a character is dropped is higher than characters being substituted and that multiple, sequential, character losses are common.

The message ETESEDEDEED arrives at the receiver and needs to be matched to one of the four candidate messages. We need to determine the correspondence between the received message and the candidate messages that maximizes the chance of properly identifying the message. Want to evaluate is the similarity of the received message to the candidate messages. Our assumption is that the candidate message with the most similarity to our received message is most likely to be the message sent.

One way to evaluate similarity is to consider all possible alignments of the received message and the candidate messages. By using a system that rewards matches and penalizes mismatches and gaps, we can assign a score to each alignment. Using the highest of those alignment scores, we find the maximum likelihood that the received message is related to each of the candidate messages. The highest of all the alignment scores tells us which message is the most likely one sent. This method has a very good chance of finding the best candidate message, but is computationally expensive because all possible alignments need to be evaluated.

Another way to evaluate find most likely message sent is to assume that the received message must have some set of characters that identically match a set of characters in the original message. The remainder of the received message should match to the source message by applying substitutions and gaps as needed. In our example, notice that received message ends in “EED.” There is one place in fourth message that EED matches exactly. There are no double or triple character strings in the received message that exactly match any position in the first three candidate messages. Ignoring spaces, if we align “EED” in the received message to the fourth message we get an alignment that is easily explained by applying the error set:

```
ENTERINGSAFEMODENEEDHELP
E-TE----S—E---DEdEED----
```

where the lower case “d” in the received message indicates a substitution error and the “-“ indicates a character loss. This second method is much faster and computationally cheaper than the first but also less sensitive because it is possible that a message

arrives that does not have two or more consecutive characters that match identically to any candidate messages.

Still another means for evaluating similarity is to compute the frequency of certain character patterns in each candidate message and then compare those frequencies to the patterns in the received message. For example, the pattern “EE” occurs once in the fourth message but not at all in the first three. “EE” also occurs once in the received message. This method is computationally inexpensive, but lacks sensitivity to the context of the characters.

This simple example makes two key points. First, there are often multiple ways to perform a similarity evaluation. Each method has advantages and disadvantages in terms of accuracy and computational cost. Second, it is not possible to know with 100% certainty whether any single method will find the right match. In some cases, a combination of methods will give us higher confidence in an answer than any individual method. Similarity comparisons provide evidence of sequence closeness only. Ultimately, investigators need to review the results and make final decisions based on their domain knowledge.

Similarity and Biology

Like messages from space, biological sequences undergo transformations that alter structure and meaning. Transformations may be the result of evolutionary processes such as species-specific variations of a protein or may be the result of errors introduced in preparing or reading the composition of sequences. Most biological similarity searches assume a simple error model. In this model, independent processes may result in the insertion, deletion, or substitution of characters in the text string that represents the residues of a peptide or nucleotide molecule.

Evolutionary mutation processes are strongly regulated by natural selection since unfavorable alterations eventually disappear. Evolutionary mutations are limited in scope and occur at a predictable rate. Like the limited set of spacecraft messages and transmission errors, restrictions on the form and rate of evolutionary mutations may be exploited in similarity comparison algorithms.

Conversely, the transformation processes associated with preparing and reading sequences tend to be random along most of the sequence length. In some situations, the transformation rate is higher at the sequence ends than in the middle of the sequence, but the nature of the transformations does not change. This knowledge too may be exploited in similarity comparison algorithms.

Why do biological similarity comparisons?

Corresponding to the modes of biology sequence variation, there are two types of sequence assessment: homology evaluation and contextual analysis. Although these evaluations have much in common, they are fundamentally asking different questions.

Homology evaluation looks for evidence that biological sequences are related but have been altered relative to each other by evolution. Orthologs are related molecules that have been changed due to speciation while paralogs are replicated molecules in the same organism that have been altered through generations of independent mutation. Homology analyses require predetermine notions of which mutations are allowed and the rate at which they can be expected to occur. These analyses therefore depend on a prior analysis of related sequences.

Contextual analyses look for the common features among sequences without concern for whether the sequences have a common ancestor. These comparisons determine whether sequences overlap or are contained within another sequences, for example. Contextual analyses are commonly part of the processing needed to join together many, small sequences into fewer, long, sequences. Contextual comparison is also used to find vector contamination in sequences, evaluate primer candidates, do biochip design, and so forth. These evaluations need to model relevant sequencing instrumentation and chemistry.

In both types of analyses, an objective function is defined that endeavors to determine the best alignment of the sequences in question. Consider two sequences: SEQUENCESEARCH and SEQUENCER. A simple evaluation of these sequences assigns one sequence to the rows of a matrix and the other sequence to the columns. Placing an asterisk in each cell for which the row and column characters match and a blank where they do not creates a diagonal of asterisks as shown in Figure 1. The longest diagonal in Figure 1 is the best alignment of the two sequences.

	S	E	Q	U	E	N	C	E	R
S	*								
E		*			*			*	
Q			*						
U				*					
E		*			*			*	
N						*			
C							*		
E		*			*			*	
S	*								
E		*			*			*	
A									
R									*
C							*		
H									

Figure 1: Comparison of Two Sequences

The simple method of Figure 1 is the basis for a powerful similarity tool called a dot plot. Dot plots provide visual evidence of similarity and may be used for comparing very large sequences. It is common to use more complex similarity comparisons because these enable computing statistical parameters associated with the probability that the sequences in question are not related. Still, these more complex methods are fundamentally creating strings of aligned sequence just as the dot plot does.

We next look into the details of homology and contextual similarity searches. We also discuss complicating factors that are important in the proper selection of algorithms and the penalty/reward systems used for a particular investigation.

Homology and similarity

Evidence of homology is sought by using a similarity objective function and set of rewards and penalties that model evolution. A comparison algorithm scores sequence similarity using the evolutionary model and also determines the likelihood that the score achieved resulted from randomly related sequences. Evolutionary models used in sequence analyses are fairly simple and based on collections of sequences that are related to each other in function and years of separation.

Consider creating an evolutionary model for transmembrane proteins that could be used to evaluate newly found proteins. A common feature in these proteins is at least one transmembrane region comprising an α-helix stretch of between 21 and 26 hydrophobic amino acid residues. While several organizations of these transmembrane regions occur, a common organization comprises seven regions connected in a serpentine. Human, mouse, rat, chicken, and bovine seven transmembrane (7TM) proteins have been found and sequenced. These orthologous sequences are separated in time by no more than the point in history that these species diverged from a common ancestor. To create a suitable evolutionary model for 7TM proteins, it suffices to create a multiple sequence alignment such that it is possible to determine what substitutions and gaps occur. A portion of one such alignment is shown in Figure 2.

```
5H1A_HUMAN/53-400   GNACVVAAIALERSLQ....NVANYLIGSLAVTDLMVSVLVLPMAALYQVL
5H1B_HUMAN/66-369   SNAFVIATVYRTRKLH....TPANYLIASLAVTDLLVSILVMPISTMYTVT
5H7_HUMAN/98-384    GNCLVVISVCFVKKLR....QPSNYLIVSLALADLSVAVAVMPFVSVTDLIG
5HT1_DROME/179-507 GNVLVCIAVCMVRKLR....RPCNYLLVSLALSDLCVALLVMPMALLYEVL
B1AR_HUMAN/75-377   GNVLVIVAIAKTPRLQ....TLTNLFIMSLASADLVMGLLVVPFGATIVVW
DADR_HUMAN/40-331   GNTLVCAAVIRFRHLR...SKVTNFFVISLAVSDLLVAVLVMPWKAVAEIAG
A1AD_HUMAN/113-402 GNLLVILSVACNRHLQ....TVTNYFIVNLAVADLLLSATVLPFSATMEVLG
D2DR_BOVIN/51-427   GNVLVCMAVSREKALQ....TTTNYLIVSLAVADLLVATLVMPWVVYLEVVG
HH2R_CANFA/35-288   GNVVVCLAVGLNRRLR....SLTNCFIVSLAITDLLLGLLVLPFSAFYQLS
5H6_RAT/43-320      ANSLLIVLICTQPALR....NTSNFFLVSLFTSDLMVGLVVMPPAMLNALYG
```

Figure 2: A portion of the multiple sequence alignment for 10 7TM proteins

Notice that the first residue is almost always a “G” but can be either an “S” or an “A.” The second character is always an “N,” and the next character can be “A,” “C,” “V,” or “L.” The dots in the alignment indicate positions in which gaps are needed to properly align the sequences.

There are several parameters that can be derived from the alignment in Figure 2. One parameter is percent identity which is the number of residues that identically match in an alignment divided by the total number of residues in the alignment. Since mutations occur at more or less a fixed rate, then any protein compared to a 7TM protein should have roughly the same percent identity as the sequences in Figure 2 have to each other.

Other parameters that can be determined from Figure 2 are the frequency of substituting given characters for each other. It is possible to derive an overall view of those frequencies and relate those to the frequencies of random substitutions. This is the approach taken in the creation of the well-known scoring matrices such as BLOSUM and PAM. It is also possible to evaluate position-dependent substitution frequencies and relate those to random substitutions. This is the approach taken in hidden Markov models and profiles.

Lastly, another parameter derived from Figure 2 is frequency and size of gaps. This information may be used to create simple affine gap models (e.g., gap open and gap extension penalties) or more specific position-dependent gap models.

Interestingly, percent identity might be higher in ortholgous sequences than in paralogous sequences. For example, human and mouse diverged around 35 million years ago meaning that orthologous human and mouse proteins may differ only by the number of successful mutations that could occur in that time period. Human and mouse orthologs therefore have approximately 90% identity to each other. Conversely, there are paralogous human proteins that have orthologs in worm. It is possible therefore that these human paralogs diverged as much as 600 million years ago. Clearly these sequences will likely be less similar to each other than mouse and human orthologs. For this reason, it is often safer to use the most sensitive similarity algorithms when looking for paralogs than for ortholog investigations because it isn’t always possible to know the evolutionary distance of orthologs.

When searching more distant homologs it is also advantageous to consider as many examples of the gene or protein as possible. This allows for more leniency in the rewards and penalties where diversity isn’t important and more stringency where diversity must be restricted. Techniques such as hidden Markov models and profiles, for example, are ideally suited for high sensitivity and specificity searches involving divergent homologies.

A caution about similarity searching and homology to remember: high degrees of similarity do not assure homology but only provide one clue which when combined with other clues might suffice to confidently declare homology. It is easy to show, for example, that small nucleotide strings such as promoter regions can have high degrees

of similarity to unrelated sequences. Most similarity search tools include post-search analyses that use sequence length and other factors to correct scores for known statistical biases.

Random transformations and similarity

Investigations requiring tolerance of random transformations generally involve nucleic acid sequences. Some investigations necessitate near-identity matching of relatively few residues, others require tolerance of moderate to high error rates over longer stretches. Near-identity similarity searching is adequately performed with lower sensitivity algorithms. In fact, in some situations regular expression searches are sufficiently accurate and fast. More sensitive and potentially more computationally intensive algorithms are needed to tolerate complex error models.

Each investigation has it's own, unique error model that is unrelated to the species involved or evolution. It is usual to define scoring systems that produce positive scores when two sequences share a minimum percent identity. The minimum percent identity required to achieve a positive match score is calculated from the match and mismatch scores:

$$\%ID = 100 * \left(1 - \frac{match}{match + |mismatch|} \right)$$

Table 1 lists representative match and mismatch scores and the associated minimum percent identity.

Table 1: Match/Mismatch Scores & Minimum Percent Identity

Match Score	Mismatch Score	% Identity
1	-1	50.0
1	-2	66.7
1	-3	75.0
1	-10	90.9
1	-15	93.8
1	-20	95.2

To illustrate computing a similarity score based on percent identity, we use the +1/-1 matrix to score the alignment:

```
A A C C T T G G G A G A C C G A T
A A T C C C G G G A G A C C T A T
+ + - + - - + + + + + + + + - + + = 13 - 4 = 9
```

Gap penalties are usually selected relative to the match/mismatch scores. It is common to set the gap open penalty between 0.5 and 5 times the match score and the gap extension penalty to be between 0.1 and 1 times the gap open penalty.

Local and global similarity

Sequence similarity may be evaluated globally or locally depending on the needs of an investigation. Global similarity requires consideration of the total lengths of the sequences under study while local similarity may consider only portions of sequences that have the best matches. Global alignments are used to find conserved regions of sequences that are already known to be homologous, for example. Local similarity is typically used in contextual investigations and in determining whether unannotated sequences might be homologous to annotated sequences.

Paracel's Similarity Search Algorithms

Authors have traditionally classified algorithms on the basis of the type of inputs used, the manner in which rewards and penalties are accessed, algorithm rigor, complexity, and so forth. We take a no less controversial, but different approach by classifying algorithms based on their utility in biological similarity searching. Detailed discussions of each algorithm are provided in the next section.

Figure 3 is a conceptual representation of the similarity search space based on the expected percent identity of the sequences being compared and the type of search being performed. The y axis in Figure 3 is a search type continuum starting with contextual analysis at the bottom and moving upward to closely related homology searching and distantly related homology searching. Equivalently, the x axis in Figure 3 is a continuum starting with highly similar sequences on the left and more dissimilar sequences on the right. Algorithms shown toward the bottom, left in Figure 3 tend to be simpler and faster than algorithms shown toward the upper right. The BLAST and Smith Waterman variants overlap each other and span a wide range of applicability. Profile variants tend to be best used in low identity, homology applications while regular expression searches are best used in high identity, contextual evaluations. BLAST and Smith Waterman are similar in that these both use position-independent residue substitution rewards and penalties. Conversely, profiles can have different rewards and penalties at each character position as a function of conserved and non-conserved residues.

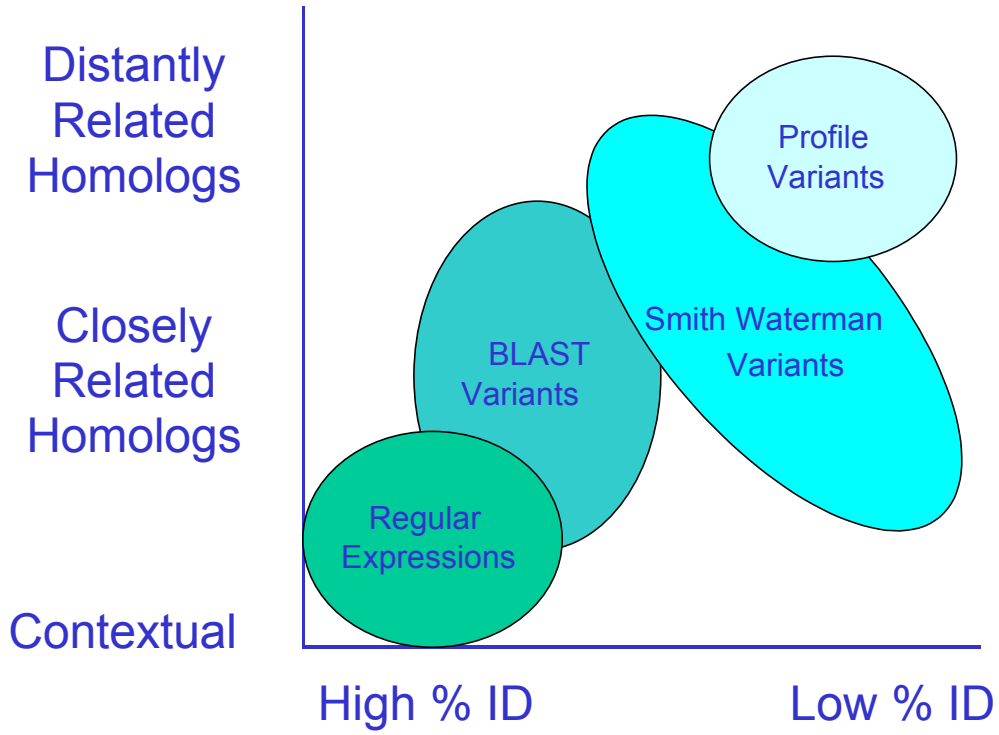

Figure 3: Search domain classification for various algorithms.

Although there are far too many applications for similarity searching than can be listed, Table 2 provides guidance on the percent identity and similarity type typical of a few common similarity investigations.

Table 2: Parameterization of Common Similarity Investigations			
Investigation	**% ID**	**Type**	**Comments**
Gene finding using ESTs	High	Contextual and homology	
Gene structure prediction using cDNA or proteins	High	Contextual and homology	Intron tolerance required
Annotation	High to low	Homology	
Primer design	High	Contextual	Generally want exact matches, no gaps.
Overlap detection	High	Contextual	Generally want nearly identical matches
Vector contamination	High	Contextual	Generally want nearly identical matches

Table 2: Parameterization of Common Similarity Investigations			
Investigation	**% ID**	**Type**	**Comments**
Simple repeats	Moderate to high	Contextual	
Ancient repeats	Moderate to low	Homology	

Paracel accelerates BLAST, Smith Waterman, and profile similarity search variants. Tables 3-5 enumerate these variants and summarizes their usage. Referring back to Figure 3, regular expression searches are performed by PHI-BLAST which is listed in Table 3 as one of the BLAST variants. It is also possible to create profiles from regular expressions and search these using a profile search variant. In some cases, regular expression searches are efficiently performed by Perl or similar scripts.

Table 3: BLAST Algorithms			
Algorithm	**Query Type**	**Database Type**	**Purpose**
BLASTN	Nucleotide	Nucleotide	Moderate sensitivity, high specificity homology and contextual searching
BLASTP	Peptide	Peptide	Moderate sensitivity, high specificity homology searching
BLASTX	Nucleotide	Peptide	Moderate sensitivity, high specificity homology searching; six frame translation of query sequences
TBLASTN	Peptide	Nucleotide	Moderate sensitivity, high specificity homology searching; six frame translation of database sequences
TBLASTX	Nucleotide	Nucleotide	Moderate sensitivity, high specificity homology searching; six frame each search of query and database
PSI-BLAST	Peptide	Peptide	Moderate to high

Table 3: BLAST Algorithms			
Algorithm	**Query Type**	**Database Type**	**Purpose**
			sensitivity, high specificity homology searching
PHI-BLAST	Regular Expression & Peptide	Peptide	Low to moderate sensitivity and specificity homology searching
MEGABLAST	Nucleotide	Nucleotide	Low to moderate sensitivity and specificity contextual searching

Table 4: Smith Waterman Algorithms			
Algorithm	**Query Type**	**Database Type**	**Purpose**
Smith Waterman Nucleotide (linear gap penalties)	Nucleotide	Nucleotide	High sensitivity and specificity contextual searching.
Smith Waterman Nucleotide (affine gap penalties)	Nucleotide	Nucleotide	High sensitivity and specificity homology searching.
Smith Waterman Nucleotide (double affine gap penalties)	Coding DNA or RNA	Genomic DNA	High sensitivity and specificity homology searching with intron tolerance.
Smith Waterman Peptide (affine gap penalties)	Peptide	Peptide	High sensitivity and specificity homology searching.
Smith Waterman Peptide (double affine gap penalties)	Peptide	Peptide	High sensitivity and specificity homology searching with long gap tolerance.
Smith Waterman frame search (affine gap)	Nucleotide	Peptide	High sensitivity and specificity homology searching; frameshift tolerant.
Smith Waterman reverse frame search (affine gap)	Peptide	Nucleotide	High sensitivity and specificity homology searching of coding regions; frameshift tolerant.

Table 4: Smith Waterman Algorithms			
Algorithm	**Query Type**	**Database Type**	**Purpose**
Smith Waterman reverse frame search (double affine gap)	Peptide	Genomic DNA	High sensitivity and specificity homology searching of coding regions; frameshift and intron tolerant.

Table 5: Profile Algorithms			
Algorithm	**Query Type**	**Database Type**	**Purpose**
Gribskov profile search	Protein profile	Peptide	High sensitivity, specificity distant homology searching.
Gribskov profile search	Nucleotide profile	Nucleotide	High sensitivity, specificity distant homology searching.
Gribskov profile frame search	Protein profile	Nucleotide	High sensitivity, specificity distant homology searching of coding regions.
Hidden Markov Model	Protein profile	Peptide	High sensitivity, specificity distant homology searching.
Hidden Markov Model	Nucleotide profile	Nucleotide	High sensitivity, specificity distant homology searching.
Hidden Markov Frame search	Protein profile	Nucleotide	High sensitivity, specificity distant homology searching of coding regions, frameshift tolerant.
Hidden Markov Frame search (double affine gap)	Protein profile	Genomic DNA	High sensitivity, specificity distant homology searching of coding regions, frameshift and intron tolerant.

Table 5: Profile Algorithms			
Algorithm	**Query Type**	**Database Type**	**Purpose**
GeneWise	Protein profile or peptide	Genomic DNA	High sensitivity, specificity distant homology searching of coding regions, frameshift and intron tolerant.

BLAST variants

Description

Basic Local Alignment Search Tool (BLAST) nucleotide, peptide and frame algorithms find multiple high scoring local similarities among sequences having moderate to high similarity. Underlying the BLAST heuristic is the idea that all true matches will consist of short stretches of identical or near-identical matches. BLAST initially extends these seed matches outward as far as possible without introducing gaps. The ungapped alignments are then combined using dynamic programming to form the final, gapped alignments. BLAST's dependence on finding short, near identity matches however, reduces it's sensitivity when many short gaps occur.

Usage

BLAST is a fast, general purpose similarity search tool that may be used in contextual and homology analyses. BLAST characteristically has good sensitivity and very good specificity meaning that it almost always finds true matches and virtually always does not report false positives. An important feature of BLAST is it's ability to report multiple local alignments between sequences. This feature allows an investigator to look for repetitive elements, domain ordering, evidence of gene transposition, and so forth.

BLASTN

BLASTN compares nucleic acid sequences to each other. BLASTN uses one match score and one mismatch score for all characters as well as an affine gap model. The default seed word size is 11 characters so sequences under investigation must be fairly similar in order for BLASTN to report any hits.

BLASTP

BLASTP is the protein equivalent of BLASTN. BLASTP permits the use of a limited number of evolution-based scoring matrices such as PAM and BLOSUM along with a limited set affine gap parameters. The default seed word size is 3 for BLASTP which means that BLASTP searches will run slower than BLASTN but have the advantage of allowing more diversity in the sequences under study.

BLASTX

BLASTX compares nucleic acid coding sequences to protein databases. The algorithm begins by translating the queries into six proteins corresponding to a three-frame forward translation and a three-frame reverse-compliment translation. The six proteins are then searched against the protein data using the BLASTP algorithm. Results are linked to the frame but there is no frame-to-frame linkage.

TBLASTN

TBLASTN compares protein queries to nucleic acid genomic or coding data. Nucleic acid data are converted into six proteins as described for BLASTX and then BLASTP is executed. As with BLASTX comparisons, TBLASTN reports results linked to the protein frame.

TBLASTX

TBLASTX compares nucleic acid coding sequences to other nucleic acid coding sequences through translated proteins. The query and data sequences are each converted into six protein frames which are then searched using BLASTP and the results are linked to the query and database frames. This search is particularly useful when nucleic acid sequences are too divergent for BLASTN to report hits even though those sequences code for similar proteins.

PSI-BLAST

Position specific iterative BLAST (PSI-BLAST) initially creates a profile from the results of a BLASTP search. Results above a threshold are aligned to each other and the residue substitution frequencies are calculated at each character position. Highly conserved positions contribute to higher match rewards and mismatch penalties than weakly conserved positions. The profile is used as the input to the next iteration and the returned results above a threshold are again multiply aligned and another profile computed. PSI-BLAST increases search sensitivity but care must be taken when selecting search results at each iteration to prevent specificity degradation. PSI-BLAST is useful in low to moderate homology evaluations.

PHI-BLAST

PHI-BLAST (Pattern-Hit Initiated BLAST) finds sequences that contain an occurrence of a regular expression that are also homologous to another sequence in the region(s) near the regular expression. In many cases, protein domains, for example, can be represented by regular expressions. The advantage of doing regular expression searches is that they are computationally inexpensive and easy to implement. The disadvantage of regular expression searches is that by themselves are not usually sufficiently specific to remove random matches. PHI-BLAST improves the specificity of regular expression searches.

MEGABLAST

MEGABLAST is used for evaluations of highly similar, large nucleotide sequence sets. MEGABLAST will combine smaller queries internally to reduce execution time, however,

the results obtained by combining queries is different than if the queries were not combined.

Smith Waterman variants

Usage

The Smith Waterman evaluation finds an optimal, local alignment of nucleotide or peptide sequences and is typically used when low to moderate sequence identity is expected. Alignments are optimal because the Smith Waterman algorithm considers all possible ways that two sequences can be matched up and reports the one having the best score. Paracel supports three gap penalty models corresponding to randomly introduced gaps, evolution-induced gaps, and introns. Additionally, like BLAST, Paracel's implementation of Smith Waterman permits reporting multiple high scoring segments of an alignment.

Sensitivity

Although the BLAST algorithms are similar to Smith Waterman, there are three significant differences that make Paracel's Smith Waterman more sensitive:

1. The BLAST algorithm begins by finding short regions of exact match between pairs of sequences. These seed regions are then extended, without gaps until the extension causes the score to drop below a threshold. Extended regions are then joined with gaps through a dynamic program. The effectiveness of a BLAST search is heavily dependent on the size of the initial seed region. Smaller regions produce better results because they are less likely to miss a true alignment but the computations are very expensive. On the other hand, longer initial word sizes make for faster searches but the results are not as good. The more evolutionary distance there is between two sequences, the greater the chance that BLAST will miss a real hit. Smith Waterman, by contrast, examines all alignments and reports an alignment having the highest score. This assures that an optimal alignment (there may be more than one) is always reported.

2. In seed processing, BLAST randomly substitutes one of the nucleic acids for "N" in DNA sequence searches. Many sequences such as genomic scaffolds or sequences masked by low-complexity filtering, can contain long strings of "N." By randomly substituting nucleic acids for "N," BLAST may fail to create a seed and therefore miss a hit. Paracel's Smith Waterman algorithm does not need to make this substitution because it uses explicitly specified scoring matrices. The reward for matches with "N" may be explicitly specified to represent the biology, chemistry, or quality of the sequences involved.

3. In DNA searches, BLAST provides a match and mismatch score parameter that is applied to all nucleic acids. This model is appropriate for evaluating differences due to sequencing error in which all modifications are equally likely, i.e., in contextual evaluations. Evaluating mismatches due to evolution requires more complicated scoring models that take into account actual differences

between related sequences. This leads to matrices that may penalize certain substitutions differently than others, and additionally, may be asymmetric to penalize changes in one direction more heavily than another. Because Paracel's Smith Waterman implementation uses explicitly specified substitution matrices for both DNA and protein searches, users can select the scoring matrix that best models the situation.

Gap options

Paracel's Smith Waterman algorithms support three gap penalty options:

1. Linear: the penalty for starting a gap is the same as the per character penalty for extending a gap. Linear gap penalties are most appropriate when modeling sequence variations due to random processes rather than evolutionary processes.

2. Affine: it is assumed that starting a gap is a more important event than extending a gap. In an affine gap model the open penalty is higher than the extension penalty. Affine gap penalties are typically used to model evolutionary variations.

3. Double affine: it is assumed that sequencing errors or minor biological variations can only reasonably account for gaps of a certain size. Gaps longer than this size tend to represent major alterations such as the introns that separate coding regions of a gene or extensions of unessential structures such as loops in some proteins. A double affine gap model enables penalizing short gaps at a different rate than longer gaps.

   Double affine consists of two sets of gap open (g_o) and gap extension (g_e) penalties and at any gap length, the algorithm chooses the smaller of the penalties. This means that the line defined by the first gap open and first gap

$$g_{o1} + g_{e1} * (l-1) = g_{o2} + g_{e2} * (l-1)$$

   extension must intersect the line defined by the second gap open and second gap extension:

   A typical cross-over length for DNA sequences is 3 to 6 nucleic acids while 2 or 3 amino acids is typical for proteins.

   The power of double affine is shown in the partial alignment below. The alignment compares a cDNA sequence from Unigene to genomic data. Notice the long gap starts with a GT at the 5' end and terminates in CAG at the 3' end which defines the classical intron donor and acceptor sites. A BLAST search might find the two pieces of this alignment if these had sufficiently high score but would be unable to correctly show the gene structure.

```
Query:     3 ATGGCGGCTGGAGGCGATCATGGTTCGCCCGACAGCTACCGCTCACCTCTTGCCTCCCGC 62
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct: 48787 ATGGCGGCTGGAGGCGATCATGGTTCGCCCGACAGCTACCGCTCACCTCTTGCCTCCCGC 48846

Query:    63 TATGCCAGCCCGGAGATGTGCTTCGTGTTTAGCGACAGGTATAAATTCCGGACATGGCGG 122
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct: 48847 TATGCCAGCCCGGAGATGTGCTTCGTGTTTAGCGACAGGTATAAATTCCGGACATGGCGG 48906

Query:   123 CAGCTGTGGCTGTGGCTGGCGGAGGCCGAGCAG-------------------------- 156
             ||||||||||||||||||||||||||||||||
Sbjct: 48907 CAGCTGTGGCTGTGGCTGGCGGAGGCCGAGCAGGTAACGGATCCCGGGCTGAGGGGCTGG 48966

Query:   156 ------------------------------------------------------------ 156
Sbjct: 48967 GCCGGGAGGGACGGGCCCGCCCCAGCACGTGCCGGGCTCTGTTCCGGGCTGGGCTTAGCC 49026

Query:   156 ------------------------------------------------------------ 156

Query:   156 ------------------------------------------------------------ 156

Sbjct: 51967 TGCTAACATGAATCAGTTTTTTTTCCTTGGTGTCACTTCATTCAAATAACTGTGACACTG 52026

Query:   156 ---------------------------------ACATTGGGTTTGCCTATCACAGATGAA 182
                                               ||||||||||||||||||||||||||
Sbjct: 52027 AGACTATTTTATTTTATTTTGCCTATTCTGCAGACATTGGGTTTGCCTATCACAGATGAA 52086

Query:   183 CAAATCCAGGAGATGAAATCAAACCTGGAGAACATA-GACTTCAAGATGGCAGCTGAGGA 241
             ||||||||||||||||||||||||||||||||||  ||||||||||||||||||||||||
Sbjct: 52087 CAAATCCAGGAGATGAAATCAAACCTGGAGAACAT-CGACTTCAAGATGGCAGCTGAGGA 52145

Query:   242 AGAGAAACGTTTACGACATGATGTGATGGCTCACGTGCACACATTTGGCCACTGCTGTCC 301
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct: 52146 AGAGAAACGTTTACGACATGATGTGATGGCTCACGTGCACACATTTGGCCACTGCTGTCC 52205

Query:   302 AAAAGCTGCAGGCATTATTCACCTTGGTGCTACTTCTTGCTATGTTGGAGACAATACTG- 361
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct: 52206 AAAAGCTGCAGGCATTATTCACCTTGGTGCTACTTCTTGCTATGTTGGAGACAATACTGT 52265
```

Smith Waterman nucleotide and peptide variants

SWN

Smith Waterman nucleotide (SWN) compares nucleic acid sequences. Paracel's implementation allows the user to specify an arbitrary match/mismatch matrix so that SWN may be used for both contextual and evolutionary comparisons. The matrix need not be symmetric to permit modeling directional substitutions.

SWP

Smith Waterman peptide (SWP) compares peptide sequences. Generally SWP is used for homology analysis and one of the evolutionary matrices, e.g., BLOSUM, is used. Unlike BLASTP, SWP does not restrict the value of permitted gap penalties.

Smith Waterman frame variants

Paracel accelerates three per-character, frameshift-tolerant, Smith Waterman style algorithms. In each of these algorithms, at each character position the score is determined by evaluating whether to stay in the current reading frame and accepting a

match/mismatch score or an amino acid insertion/deletion (indel) or to jump to another reading frame and incur a frameshift penalty along with a match/mismatch score. This contrasts to the equivalent BLAST search types in which six static protein translations corresponding to three forward frames and three reverse frames are used in the comparison. Paracel's frame search variants tolerate frameshifts that are most often the result of sequencing errors and produce longer meaningful alignments then can be produced by BLAST.

SWX

Paracel's frame search compares nucleic acid query sequences to protein data. This search is used to find putative homologous proteins for newly sequenced ESTs, RNAs, and cDNAs. An independently adjustable frameshift penalty may be set to reflect the overall quality of the nucleic acid sequences. Additionally, this algorithm uses protein scoring matrices that can be chosen to reflect the evolutionary distance between the nucleic acid sequences and the organisms represented in the protein database. An affine gap penalty is generally used to model evolutionary variations.

TSWN

Searching a peptide sequence against nucleic acid coding regions is performed with Paracel's reverse frame algorithm. This comparison allows a user to annotate unknown peptide sequences by comparing them to databases of nucleic acid coding regions or to locate putative genes with known proteins. An independently adjustable frameshift penalty is available to model the possibility that a sequencing error in the nucleic acid data has occurred. Protein scoring matrices are used along with affine gap penalties to model evolutionary variations. Double affine gap penalties may be used to evaluate gene structure.

TSWX

Lastly, Paracel offers a double frame nucleic acid to nucleic acid comparison at the protein level. This search allows for frameshifts at each character of both nucleic acid sequences. This search is useful for comparing homologous coding regions that are sufficiently separated by evolution to have differing codon usage.

Profile variants

A profile is a mathematical summary of the alignment of multiple, related nucleic acid or protein sequences. A profile is built by scanning down each column of the multiple sequence alignment to determine the frequency of each character in the column. These position-dependent character frequencies are normalized by the frequency of each character in the multiple alignment to produce an odds ratio for the substitution of one character for another.

Profile searches enable a comparison of database sequences with the family represented by the profile. The position-specific nature of these searches facilitate a comparison of the critical features characteristic of the family. For example, a multiple alignment of a protein or nucleotide family will have highly conserved regions

interspersed with non-conserved regions. The profile gives high scores to matches of a query within the conserved areas and correspondingly very low scores for mismatches. In non-conserved regions match and mismatch scores have lower positive or negative impact to reflect the reduced contribution of those regions to the function or structure of the protein or nucleic acid sequence.

Gribskov profile and profile frame

Paracel's *profile* function is implemented in the GeneMatcher as a position-specific Smith Waterman. This means that each position of a query will have it's own set of scoring parameters as determined by the content of the profile. Paracel's profile search takes protein or nucleic acid profile inputs in GCG format and compares these to a protein or nucleic acid databases respectively. Results may be reported either as a search of database entries by the profiles or as a search of profiles by the database entries.

The profile frame search variant enables searching protein profiles against nucleic acid coding regions using a per-character frameshift tolerant algorithm. This algorithm is essentially a position-specific version of Paracel's TSWN algorithm discussed previously.

Hidden Markov Model and Hidden Markov Model Frame Searches

A hidden Markov Model (HMM) is a profile trained from a multiple sequence alignment in a manner analogous to the profile discussed above. While profiles assume a relatively simple model of evolution, HMMs allow for more options which often enhances the sensitivity searches. In particular, at each position in an alignment, HMMs have an explicitly computed probability associated with starting or continuing a gap. Additionally, at each insertions point, an HMM explicitly models the probabilities associated with adding specific amino acids. HMMs are best used for homology searching of evolutionary distant sequences.

Paracel's HMM search currently accepts PFAM, Bucher's Generalized Profile, and SAM input formats. Nucleic acid models and protein models may be used to search respectively against nucleic acid or protein databases.

Paracel's HMM frame algorithm is analogous to the profile frame algorithm in that it allows a search of nucleic acid sequences by protein HMMs. This search also uses a per-character frameshift tolerant algorithm. HMM frame searches are used to search coding nucleotides or by using a double affine option, genomic data.

GeneWise

GeneWise is an intron and frameshift tolerant search of genomic data by proteins or protein HMMs. GeneWise facilitates determination of gene structures by expanding the HMM concept to include transitions to and from intron states. This means that during the search process, the algorithm evaluates whether the best score is achieved by staying in a given reading frame and accepting a match/mismatch score, changing

reading frames and incur a frameshift penalty, accepting an amino acid indel, starting an intron if one hasn't been started, or by exiting and intron if one has been started.

Paracel's *genewise* search uses a GU (GT) pattern at the 5' end to signal the possible beginning of an intron and an AG pattern at the 3' end to signal the possible end of an intron. These patterns do not force intron start and end but rather enable the algorithm to consider whether a start or end produces the best score result.

GeneWise is similar to Paracel's HMM frame search with the additional benefit of tolerating long introns. It is also similar to Paracel's double affine Smith Waterman search with the additional benefit of using protein HMMs to better model the important features of a protein family.