

PatternHunter 2.0 User Manual



For use with the PatternHunter DNA
and protein homology search tool.
Copyright 2005
Bioinformatics Solutions Inc.
All rights reserved.

BIOINFORMATICS SOLUTIONS INC

PatternHunter 2.0 User's Manual

Prepared By: Iain Rogers, Derek Kisman
© Bioinformatics Solutions Inc.
145 Columbia St. West Suite 2B
Waterloo, Ontario Canada N2L 3L2
Phone 519-885-8288 • Fax 519-885-9075

INTRODUCTION.....	2
INTRODUCTION TO PATTERNHUNTER 2.0	2
GETTING STARTED WITH PATTERNHUNTER 2.0.....	3
PACKAGE CONTENTS	3
PLATFORM AND JAVA VIRTUAL MACHINE	3
<i>Memory issue:</i>	3
<i>Native Compiler Issue</i>	4
DOWNLOADING AND REGISTERING.....	4
INPUT.....	5
PATTERNHUNTER FOR DNA INPUT	5
TRANSLATED PATTERNHUNTER INPUT	6
PATTERNHUNTER PARAMETERS	7
PARAMETERS: PATTERNHUNTER FOR DNA	7
PARAMETERS: TRANSLATED PATTERNHUNTER	12
SEARCH STRATEGY	17
PATTERNHUNTER FOR DNA.....	17
TRANSLATED PATTERNHUNTER'S.....	18
SCORING SCHEMES.....	19
ADJUSTING PATTERNHUNTER FOR DNA'S SCORING SCHEME	19
PATTERNHUNTER FOR DNA'S SCORING REQUIREMENTS.....	19
ADJUSTING TRANSLATED PATTERNHUNTER'S SCORING SCHEME.....	20
ADVANCED PATTERNHUNTER USAGE.....	21
PREPARING TO SEARCH THROUGH A LARGE DATABASE	21
RESULT GROUPING AND PROCESS BATCHING	21
ADJUSTING SENSITIVITY AND SPEED.....	23
COMMAND LIST AND DEFAULTS.....	25
PATTERNHUNTER FOR DNA.....	25
TRANSLATED PATTERNHUNTER.....	26
FREQUENTLY ASKED QUESTIONS.....	28
ABOUT BIOINFORMATICS SOLUTIONS INC.....	31
PATTERNHUNTER SOFTWARE LICENSE AGREEMENT	32

Introduction

Introduction to PatternHunter 2.0

PatternHunter is the most reliable homology search tool.

PatternHunter is a Java program for homology search. It consists of two components. PatternHunter for DNA and Translated PatternHunter for protein searches.

PatternHunter for DNA finds all "approximate repeats" and "approximate reverse complement repeats" in one DNA sequence or between a pair of DNA sequences. Repeats are local, gapped, alignments scored by match, mismatch, gapopen and gapextend parameters. PatternHunter outputs all repeats ranked by score and additionally outputs an Encapsulated Postscript File showing them as a dot plot. PatternHunter does what Blastn and MegaBlast do, except with much greater speed at the same sensitivity because it used several new and proprietary technologies, as documented in the PatternHunter paper.

Translated PatternHunter finds protein-level homologies between DNA or Protein sequences. These "alignments" between two sequences consist of paired groups of amino acids with a high level of similarity according to a scoring matrix (such as BLOSUM62). Gaps can be found in these homologies, and are penalized using gapopen, gapextend, and frameshift parameters. Translated PatternHunter outputs all repeats ranked by score. tPH encapsulates the functionality of blastp, tblastn, blastx, and tblastx in an integrated suite. tPH also supports gaps and DNA-level frameshifts in alignments, which are missing from the latter three tools. Since genuine homologies often include gaps, this makes tPH a much more accurate homology-searching tool.

Getting started with PatternHunter 2.0

Everything we need to know from the beginning and step by step.

This section of the manual provides information about system requirements and where to go to get help setting up PatternHunter.

Package contents

The PatternHunter package should contain:

- This manual
- phn.jar and tph.jar -- PatternHunter's executable program files
- license information
- A read me file

Platform and Java Virtual Machine

Translated PatternHunter is written in Java and runs on any platform with a Java VM. For optimal performance, it is important to consider the following JVM issues:

Memory issue:

PatternHunter is written in Java and runs on any computer platform with a Java VM. A java VM is a program that supports the execution of Java programs like PatternHunter. You usually can obtain a free Java VM. The maximum memory that a java application can use not only depends on the size of the physical memory on your computer, but also depends on the Java VMs.

GETTING STARTED

Here, we list several popular Java VMs with the memory limitation in the following table.

Platform Java VM Maximum memory support	
Windows (XP, 2000, NT ..)	Sun VMs 2G
Linux for X86 Sun and IBM VMs	2G
Compaq Alpha server Compaq fastVM for 64-bit	at least 32G

Note that the maximum memory support assumes that you have enough physical memory. If you do not have enough physical memory, the maximum memory PatternHunter can use is no more than what you have.

For Compaq's Alpha server, Compaq fastVM must be used to run PatternHunter. The limitation on the array size in the ordinary Compaq VM makes it impossible for PatternHunter to run.

Native Compiler Issue

JIT and HotSpot are technologies that allow the JVM to compile Java bytecodes into a native executable, then run the executable at run time. PatternHunter was designed to take advantage of this technology to achieve high run time speed. Most updated JVMs come with JIT or HotSpot. However some older JVMs do not have JIT or HotSpot, or come with old version of JIT or HotSpot which will not optimize PatternHunter to maximum speed performance. Running PatternHunter on JVMs without JIT or HotSpot will dramatically decrease the running speed by a factor of 10. Make sure that your JVM is JIT or HotSpot enabled in order to get the maximum performance.

Downloading and registering

This process is best illustrated in the PatternHunter getting started video. Please view the video at:

http://www.bioinformaticssolutions.com//movies/ph/two/PH_QuickStart.html

Input

PatternHunter for DNA Input

Normally, PatternHunter accepts two sets of sequences as input, known as the Query and Subject sequences. The Query and Subject can be single sequences, each specified by one input file in FASTA format. Alternately, multiple sequences can be compared together; they may be specified by listing multiple files, or by database files containing multiple sequences in FASTA format (or a combination). PatternHunter will compare all possible pairs of Query and Subject sequences for approximate matches.

PatternHunter can also compare a single sequence against itself, avoiding trivial and duplicate matches. To do this, simply leave out the Subject parameter.

By default, Query sequences will be compared one by one against the Subject database, with the most significant Subject sequences listed first. This is Blast's behaviour. This can be changed with the `-db` parameter.

Apart from the standard nucleotide codes A,C,G,T (or a,c,g,t; PatternHunter ignores case unless masking is specified), one can also use the standard nucleotide subset codes:

Alignment scoring for these codes is based on expected score, assuming each nucleotide in the subset occurs with equal probability.

$M = \{A, C\}$

$R = \{A, G\}$ $S = \{C, G\}$

$W = \{A, T\}$ $Y = \{C, T\}$ $K = \{G, T\}$

$V = \{A, C, G\}$ $B = \{C, G, T\}$

$H = \{A, C, T\}$

$D = \{A, G, T\}$

$N = \{A, C, G, T\}$

PATTERNHUNTER PARAMETERS

Any other non-whitespace letters are flagged but otherwise ignored.

Translated PatternHunter Input

Translated PatternHunter input is the same as PatternHunter for DNA input, with the exception that the Query and Subject may each consist of either DNA or Protein sequences, allowing you to run Translated DNA-DNA, Translated DNA-Protein, or Protein-Protein searches.

PatternHunter Parameters

A guide to the most powerful homology search software

This section of the manual will provide in depth explanation of the parameters that can be used with PatternHunter. The first part deals with **PatternHunter for DNA** (PHn). The second part refers to **translated PatternHunter** (tPH). Many parameters mean the same thing in both PHn and tPH. However, since tPH and PHn have different functionalities, their parameter sets are slightly different. Please take note.

Parameters: PatternHunter for DNA

-i *Query sequence(s)*

A file or list of files containing the sequence or sequences to be compared as the Query. If a file contains multiple sequences, all will be used.

-j *Subject sequence(s)*

A file or list of files containing the sequence or sequences to be compared as the Subject. If a file contains multiple sequences, all will be used. This parameter should be omitted if one desires to compare a single Query sequence with itself.

-o *Alignment output file*

The match results will be stored in this file. If omitted, results will be printed to the screen (standard output). For each significant match found, PatternHunter outputs the best top scoring alignments in textual form. Each alignment is preceded by a tag like

```
>1:(628409,430579)<8636,8642>4374 E=0.0
```

PATTERNHUNTER PARAMETERS

which identifies the starting position in the first sequence, then in the second sequence, the lengths in the first and second sequences, the alignment score, and finally the alignment expect value. The position in the first sequence can be negative, which indicates a match found in the reverse complement strand. Conceptually, the complement of the base at position i occurs at position $-1-i$. (also see option `-b`)

-db *Process multiple input sequences at once*

You can choose to process all the input Query sequences concurrently, or all the Subject sequences, or both, or neither. When sequences are processed together like this, they must all fit in memory, but (especially for Subject sequences) performance will be improved. In addition, when multiple sequences are compared at the same time, PatternHunter will report the results of the "best" sequences first (in terms of highest-scoring alignment).

For database vs. database matching, you may wish to use `-db 3` to have the most similar pair of sequences reported first. Also, if you are running many small Query sequences against a Subject database that is too large to fit into memory (see option `-m`), `-db 3` will speed up the search significantly. Please see the advanced PatternHunter usage for a more in-depth discussion.

-s *Strand*

By default PatternHunter will look for matches using both the top and bottom (reverse complement) strands of the Query sequence. You can restrict this using this parameter. Use 1 to search only the top strand, 2 to search only the bottom strand 3 to search both.

-a *Number of processors*

By default PatternHunter is a single-threaded application. If you are running on a system that has multiple CPUs, this option will allow PatternHunter to take advantage of them and run significantly faster.

-mi -mj *Mask lower case bases*

If you have processed your sequence file to mask certain areas (such as those with high redundancy), use this option. PatternHunter will not consider matches between nucleotides represented by lower-case letters. `-mi` allows masking of the Query sequence, and `-mj` of the Subject sequence.

-phmaski -phmaskj *Run PatternHunter's repeat masker*

PatternHunter can automatically mask out tandem repeats and large clusters of highly similar (redundant) segments from your sequences. The repeat masker only needs to run once for any given sequence file; it will store its results in the same directory as that

PATTERNHUNTER PARAMETERS

file, including a readable text file with a .phmask extension that contains reports of the repeats and clusters found. -phmaski will run the repeat masker on the Query sequence, -phmaskj on the Subject.

-maskmatch *Count masked matches towards score*

Masked bases are represented in PatternHunter's output as lowercase letters, and do not count towards the final score of the alignments they are in. If you use this option, then masked bases will be scored normally. However, they will still be ignored during initial hashing and searching for seed hits.

-N *Number of alignments to output*

For each Query/Subject sequence pair, PatternHunter will output the best N alignments it has found. The default is also the maximum.

-Ns *Number of sequences to output*

Specifies how many of the best Query/Subject sequence pairs (in order of highest-scoring alignment found between them) PatternHunter will report.

-O *Final alignment optimization*

The alignments that PatternHunter outputs are sometimes not "locally optimal": they are good alignments, but their score may be slightly increased with small gapping adjustments. This parameter tells PatternHunter to run a final Dynamic Programming pass on the alignments to find this locally optimal score, at a small additional cost in memory and time. If knowing the broad location of homologies is your only goal, this final pass is unnecessary.

The time and space required depends on the level of optimization desired. 16, for example, will find the locally optimal alignments in most cases. 64 will almost always find them.

-b *Output alignments in BLAST format*

By default, PatternHunter will output in a style similar to BLAST. If you prefer the original PatternHunter output format, use this option to print alignments in that style.

-e *Maximum alignment expectation value*

No sequences with a higher expect value than this will be reported. The expect value is a rough approximation of how significant an alignment is; between two completely random sequences, you would expect to see roughly 10 alignments with expect value 10.

PATTERNHUNTER PARAMETERS

-As *Minimum alignment score*

Alignments scoring below this value will not be considered significant, and PatternHunter will not report them.

-r *Reward for a nucleotide match*

Specifies the score for a positive nucleotide match in an alignment. (see Scoring below)

-q *Penalty for a nucleotide mismatch*

Specifies the score penalty for a negative nucleotide match in an alignment. (see Scoring below)

-G *Cost to open a gap*

Specifies the base penalty for each gap opened in an alignment. Since gaps are uncommon, but tend to stretch across multiple nucleotides, this should be of a higher magnitude than the extension penalty below. (see Scoring below)

-E *Cost to extend a gap*

Specifies the base penalty for each nucleotide a gap stretches across. (see Scoring below)

-W *Weight of model*

Number of bases collected near each position for the initial exact match search (using a hashtable). While programs like BLAST collect successive bases, PatternHunter collects nonsuccessive bases within a small window. The default, weight 11, model is `bbb*b**1b*b**bb*bbb`, where 'b' denotes an included base and '*' an excluded base (don't care). Higher weights reduce search sensitivity, and use exponentially more memory, but improve speed. The maximum weight supported is 15, which may require up to 1 GB of memory.

-multi *Simultaneous seed searching*

Using this parameter you can run PatternHunter's search using multiple seed models at once. This can significantly increase sensitivity with less cost in speed than reducing the seed weight. If you have sufficient memory, this is the best way to run highly sensitive searches with PatternHunter. Note that you can specify which range of seeds you want to use, but the easiest way to use this parameter is just to specify how many seeds you want to use (eg 2, 4, 8, or 16).

-coding *Use coding-region seeds*

PATTERNHUNTER PARAMETERS

PatternHunter uses pregenerated spaced seeds calculated to be as sensitive as possible for standard DNA searches. However, the distribution of homologies in coding regions is different (for instance, the third basepair of a codon is often unimportant); PatternHunter has seeds optimized to take advantage of this. Activate this parameter to use them.

-model *Explicit model*

If you want to specify the model for the search seed explicitly, specify it in binary using this parameter. A 1 specifies a required match, a 0 a "don't care". Note that the default PatternHunter models for each weight should be fine for most users.

-H *Number of hits to trigger extension*

By default PatternHunter will wait until two exact matches (using the model specified above) have been found close together before considering an area in more detail (extending to an HSP). This improves selectivity and hence speed at the cost of sensitivity. To instead extend all exact matches to HSPs, set this to 1.

-h *Number of hits to trigger local extension*

When searching for small local hits around a discovered HSP (see Search Strategy), PatternHunter will consider this number of hits to be significant and worthy of further expansion.

-L *Length of maximum gap*

HSPs that are at most a given number of diagonals apart are candidates for connection by a gap. PatternHunter uses the equivalent of Dynamic Programming to find the optimal way of connecting HSPs together. PatternHunter will never consider gaps longer than this value. (If a gap were to extend longer, it would require two very strong alignments on either side to make up for the penalty; thus they would probably be found and reported independently).

-s *Minimum HSP score*

After finding the maximal local alignment, Segment Pairs with scores below this minimum are considered chance occurrences and ignored. The remaining Highscoring Segment Pairs (HSPs) are used to build gapped alignments.

-l *Minimum local HSP length*

When searching the vicinity of an HSP for local matches, PatternHunter will consider Segment Pairs below this length to be chance occurrences. (This is weaker than the score requirement above)

PATTERNHUNTER PARAMETERS

-t *Maximum distance between HSPs*

When connecting HSPs (see option -L), PatternHunter will not consider HSPs further apart (not counting gaps) than this value.

-m *Size limit on database partitions*

If your Subject sequence is too large for the amount of memory you're using, PatternHunter may automatically "partition" the database, reading only part of it into memory at once. This reduces memory usage at some cost of speed. This parameter specifies how large the database must be before this occurs.

Note that, when searching multiple Query sequences on a partitioned database, PatternHunter may end up hashing the database multiple times, leading to poor performance. Use the parameter -db 3 to fix this, by running all of the Query sequences at once. See advanced PatternHunter usage for details.

-P *Hide progress meter*

Use this parameter to hide PatternHunter's percentage completion meter.

Parameters: Translated PatternHunter

-i *Query sequence(s)*

A file or list of files containing the DNA sequence or sequences to be compared as the Query. If a file contains multiple sequences, all will be used.

-j *Subject sequence(s)*

A file or list of files containing the DNA sequence or sequences to be compared as the Subject. If a file contains multiple sequences, all will be used.

-ip *Query protein sequence(s)*

A file or list of files containing the Protein sequence or sequences to be compared as the Query. If a file contains multiple sequences, all will be used. Note that -i and -ip should not be used together; the Query should be either all DNA, or all Protein, never a mix of the two.

-jp *Subject protein sequence(s)*

A file or list of files containing the Protein sequence or sequences to be compared as the Subject. If a file contains multiple sequences, all will be used. Note that -j and -jp should not be used together; the Subject should be either all DNA, or all Protein, never a mix of the two.

-o *Alignment output file*

PATTERNHUNTER PARAMETERS

The match results will be stored in this file. If omitted, results will be printed to the screen (standard output). For each significant match found, PatternHunter outputs the best top scoring alignments in textual form. Each alignment is preceded by a tag like

```
>1:(628409,430579)<8636,8642>4374 E=0.0
```

which identifies the starting position in the first sequence, then in the second sequence, the lengths in the first and second sequences, the alignment score, and finally the alignment expect value. The position in the first sequence can be negative for DNA sequences, which indicates a match found in the reverse complement strand. Conceptually, the complement of the base at position i occurs at position $-1-i$. (also see option -b)

-db *Process multiple input sequences at once*

You can choose to process all the input Query sequences concurrently, or all the Subject sequences, or both, or neither. When sequences are processed together like this, they must all fit in memory, but (especially for Subject sequences) performance will be improved. In addition, when multiple sequences are compared at the same time, PatternHunter will report the results of the "best" sequences first (in terms of highest-scoring alignment).

The default behaviour (iterating through Query sequences one by one) is equivalent to Blast. For database vs. database matching, you may wish to use -db 3 to have the most similar pair of sequences reported first. See advanced PatternHunter usage for a more in depth discussion.

-s *Strand*

By default PatternHunter will look for matches using both the top and bottom (reverse complement) strands of any DNA sequences. You can restrict this using this parameter. Use 1 to search only the top strand, 2 to search only the bottom strand 3 to search both.

-a *Number of processors*

By default PatternHunter is a single-threaded application. If you are running on a system that has multiple CPUs, this option will allow PatternHunter to take advantage of them and run significantly faster.

-mi -mj *Mask lower case bases*

If you have processed your sequence file to mask certain areas (such as those with high redundancy), use this option. PatternHunter will not consider matches between nucleotides represented by lower-case letters. -mi allows masking of the Query sequence, and -mj of the Subject sequence.

-M *Scoring matrix file*

PATTERNHUNTER PARAMETERS

BLOSUM62 is the default amino acid scoring matrix for tPH, but you may specify different matrices using this parameter. Simply point to the filename of the matrix you wish to use - PatternHunter can read BLAST-compatible matrix files. Other common scoring matrices include BLOSUM45, BLOSUM80, and PAM 70 (or you can edit your own).

-N *Number of alignments to output*

For each Query/Subject sequence pair, PatternHunter will output the best N alignments it has found. The default is also the maximum.

-Ns *Number of sequences to output*

Specifies how many of the best Query/Subject sequence pairs (in order of highest-scoring alignment found between them) PatternHunter will report.

-O *Final alignment optimization*

The alignments that PatternHunter outputs are sometimes not "locally optimal": they are good alignments, but their score may be slightly increased with small gapping adjustments. This parameter tells PatternHunter to run a final Dynamic Programming pass on the alignments to find this locally optimal score, at a small additional cost in memory and time. If knowing the broad location of homologies is your only goal, this final pass is unnecessary.

The time and space required depends on the level of optimization desired. 16, for example, will find the locally optimal alignments in most cases. 64 will almost always find them.

-b *Output alignments in BLAST format*

By default, PatternHunter will output in a style similar to BLAST. If you prefer the original PatternHunter output format, use this option to print alignments in that style.

-e *Maximum alignment expectation value*

No sequences with a higher expect value than this will be reported. The expect value is a rough approximation of how significant an alignment is; between two completely random sequences, you would expect to see roughly 10 alignments with expect value 10.

-As *Minimum alignment score*

Alignments scoring below this value will not be considered significant, and PatternHunter will not report them.

-G *Cost to open a gap*

Specifies the base penalty for each gap opened in an alignment. Since gaps are uncommon, but tend to stretch across multiple amino acids, this should be of a higher magnitude than the extension penalty below. (see Scoring below)

-E *Cost to extend a gap*

PATTERNHUNTER PARAMETERS

Specifies the base penalty for each amino acid a gap stretches across. (see Scoring below)

-F *Cost for a frameshift*

Some gaps also introduce a frameshift to an alignment, by omitting individual DNA bases. This represents the one-time penalty that is added to a frameshifted gap, and is often quite high. (see Scoring below)

-W *Weight of model*

Number of amino acids collected near each position for the initial high- similarity search (using a hashtable). While programs like BLAST collect successive amino acids, PatternHunter collects nonsuccessive amino acids within a small window.

The default, weight 5, model is 1101011, where '1' denotes an included amino acid and 0 an ignored one. If you change the model weight, be sure to choose an appropriate tolerance. (see Tolerance below)

-multi *Simultaneous seed searching*

Using this parameter you can run Translated PatternHunter's search using multiple seed models at once. This can significantly increase sensitivity at a reasonable cost in speed. If you have sufficient memory, this is the best way to run highly sensitive searches with tPH. Note that you can specify which range of seeds you want to use, but the easiest way to use this parameter is just to specify how many seeds you want to use (eg 1, 2, or 4). By default, tPH uses the maximum 4 seeds (but this can be reduced if memory usage is too high).

-model *Explicit model*

If you want to specify the model for the search seed explicitly, specify it in binary using this parameter. A 1 specifies a significant amino acid, a 0 a "don't care". Note that the default PatternHunter models for each weight should be fine for most users.

-T *Tolerance of model*

This represents how closely a seed must match sections of the query and subject sequence before a "hit" is triggered and the area is expanded in more detail. If the total score between the amino acids under the seed reaches this tolerance (or greater), a hit is found. Setting different model tolerances is the easiest way to tune tPH's speed vs. sensitivity. For weight 5 seeds, a reasonable range is from 16-25 (from slow to fast, high sensitivity to low); for weight 4, 13-20; for weight 3, 10-15; and so on.

-H *Number of hits to trigger extension*

By default tPH will wait until two highly similar matches (using the model and tolerance specified above) have been found close together before considering an area in more detail (extending to an HSP). This improves selectivity and hence speed at the cost of sensitivity. To instead extend all hits to HSPs, set this to 1.

-L *Length of maximum gap*

PATTERNHUNTER PARAMETERS

HSPs that are at most a given number of diagonals apart are candidates for connection by a gap. PatternHunter uses the equivalent of Dynamic Programming to find the optimal way of connecting HSPs together. PatternHunter will never consider gaps longer than this value. (If a gap were to extend longer, it would require two very strong alignments on either side to make up for the penalty; thus they would probably be found and reported independently). Note that this value is in terms of DNA bases; for Protein-Protein searches, divide this by 3.

-s *Minimum HSP score*

After finding the maximal local alignment, Segment Pairs with scores below this minimum are considered chance occurrences and ignored. The remaining Highscoring Segment Pairs (HSPs) are used to build gapped alignments.

-l *Minimum local HSP length*

When searching the vicinity of an HSP for local matches, Translated PatternHunter will consider Segment Pairs below this length to be chance occurrences. (This is weaker than the score requirement above)

-t *Maximum distance between HSPs*

When connecting HSPs (see option -L), Translated PatternHunter will not consider HSPs further apart (not counting gaps) than this value.

-P *Hide progress meter*

Use this parameter to hide PatternHunter's percentage completion meter.

Search Strategy

PatternHunter's search provides accurate results and fast!

This section of the manual briefly describes PatternHunter's search strategy. It is a good idea to familiarize ourselves with this strategy. With this understood, we can adjust scoring, extension and hit parameters with confidence. PatternHunter for DNA and Translated PatternHunter have very similar search strategies. But there are few key differences. Take note.

PatternHunter for DNA

The fundamental unit of the PatternHunter algorithm is the High-scoring Segment Pair (HSP). An HSP consists of two sequence fragments of arbitrary but equal length whose alignment is locally maximal and for which the alignment score meets or exceeds a threshold or cutoff score. A set of HSPs is thus defined by two sequences, a scoring system, and a cutoff score; this set may be empty if the cutoff score is sufficiently high. The sensitivity of the program can be adjusted via the PatternHunter parameters W and H ; selectivity of the program can be adjusted via the cutoff score. The task of finding HSPs begins with identifying short words in the two sequences that match according to a "model", which specifies which positions in the word must match. The number of positions that must match is called the model weight. These initial neighborhood word hits act as seeds for initiating searches to find longer HSPs containing them. The word hits are extended in both directions along each sequence for as far as the cumulative alignment score can be increased. Extension of the word hits in each direction are halted when: the cumulative alignment score falls off by the quantity X from its maximum achieved value; or the end of either sequence is reached. Besides the global hits using the user specified model, PatternHunter also finds local hits in limited size regions to the left of any HSP found, using a very sensitive weight 3 model.

Translated PatternHunter's

The fundamental unit of the PatternHunter algorithm is the High-scoring Segment Pair (HSP). An HSP consists of two sequence fragments of arbitrary but equal length whose alignment is locally maximal and for which the alignment score meets or exceeds a threshold or cutoff score. A set of HSPs is thus defined by two sequences, a scoring system, and a cutoff score; this set may be empty if the cutoff score is sufficiently high. The sensitivity of the program can be adjusted via the PatternHunter parameters W, T, and H; selectivity of the program can be adjusted via the cutoff score. The task of finding HSPs begins with identifying short words in the two sequences that match very well according to a "model", which specifies which positions in the word are significant. The number of positions that are significant is called the model weight. The score that must be exceeded before a match is identified is called the "tolerance". These initial neighborhood word hits act as seeds for initiating searches to find longer HSPs containing them. The word hits are extended in both directions along each sequence for as far as the cumulative alignment score can be increased. Extension of the word hits in each direction are halted when: the cumulative alignment score falls off by a certain amount from its maximum achieved value; or the end of either sequence is reached. Besides the global hits using the user specified model, Translated PatternHunter also finds local HSPs in limited size regions to the left of any HSP found, using a dynamic programming algorithm.

Scoring Schemes

PatternHunter's scoring is 100% customizable.

This section of the manual is a discussion of PatternHunter's scoring scheme, how to adjust it, and the constraints on values used. Adjusting scoring may not be necessary for all users, but if we chose to do so, we should be very careful. The scoring scheme is fundamental, and will drastically affect results.

Adjusting PatternHunter for DNA's Scoring Scheme

The `-r` parameter sets the reward score (R) for a pair of matching residues; the `-q` parameter sets the penalty score (Q) for mismatching residues. R and Q must be positive and negative integers, respectively. The relative magnitudes of R and Q determines the number of nucleic acid PAMs (point accepted mutations per 100 residues) for which they are most sensitive at finding homologies. Higher ratios of R:Q correspond to increasing nucleic acid PAMs (increased divergence). The default values for R and Q, respectively 1 and -1, having a ratio of 1.0, correspond to about 30 nucleic acid PAMs.

PatternHunter for DNA's Scoring Requirements

Regardless of the scoring scheme employed, two stringent criteria must be met. First, given the residue composition for the two sequences the alignment score expected for any randomly selected pair of residues (one from each sequence) must be negative. Second, given the sequence residue compositions and the scoring scheme, a positive score must be possible to achieve. For instance, the match reward score must have a positive value; and given the assumption that the 4 nucleotides A, C, G and T are represented at equal 25% frequencies, a wide range of value combinations for R and Q are precluded from use -- namely those combinations where the magnitude of the ratio R:Q is greater than or equal to 3.

Adjusting Translated PatternHunter's Scoring Scheme

Translated PatternHunter's scoring scheme may be adjusted. This is done by pointing PatternHunter at a new scoring matrix using the **-M** parameter. PatternHunter can BLAST compatible matrix files. Other common scoring matrices include BLOSUM45, BLOSUM80, and PAM 70 (or you can edit your own).

Advanced PatternHunter Usage

When the application is very time sensitive, we must adjust PatternHunter to get the best results on our first try.

T Depending on our project, tweaking PatternHunter's parameters can significantly improve the presentation and applicability of our data, as well as PatternHunter's run time. The following presents a discussion of parameters that may be useful when comparing large amounts of data – when getting it right the first time becomes crucial to efficiency.

Preparing to Search Through a Large Database

PatternHunter can search through large databases filled with sequences. That is to say, the query and/or subject of the search can contain many, many sequences. PatternHunter will load the database into memory to perform this comparison. But, if the database is too big to fit into our physical memory, PatternHunter will internally partition the database.

```
Java -Xmx512m -jar phn.jar -i query.fna -j subject.fna -m 64
```

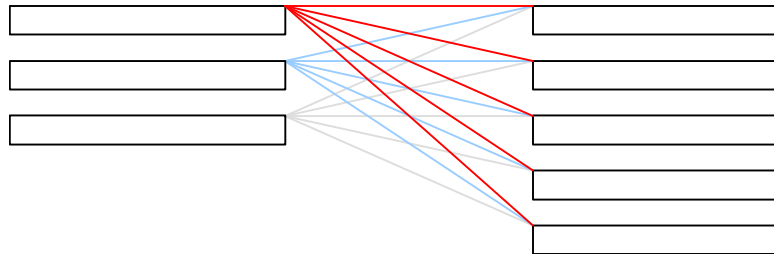
Will tell PHn to partition the database into chunks that are 64 megabytes each. Our machine's physical memory needs to be approximately 5-8 times larger than the partition size. tPH does not partition databases.

Result grouping and process batching

We can choose to process sequences (whether they be subject or query sequences) all at once, or one after the other. This may seem like a fine distinction, but it makes a big difference to the efficiency with which PatternHunter runs, and to the resulting output. PatternHunter presents us with four options. The default setting

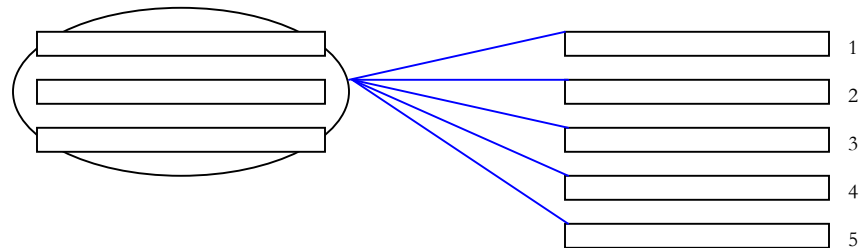
ADVANCED PATTERNHUNTER USAGE

(using the `-db 3` parameter) is the most highly recommended, as it provides nicely grouped results, and faster processing.



Option zero (-db 0):

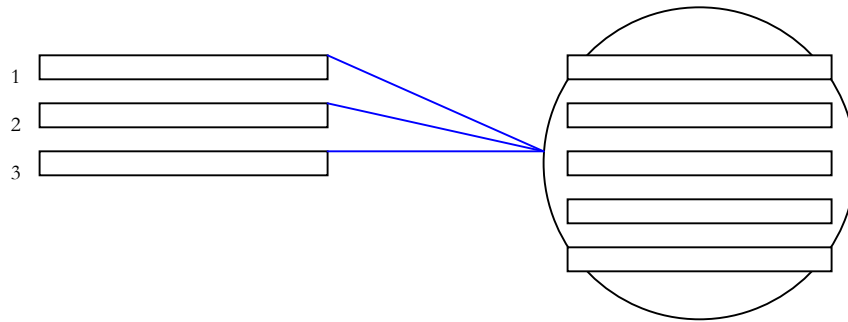
PatternHunter makes a complete pass through all the sequences in the database for each query sequence. Results are output as: All the homologies for query one in order, followed by the homologies for query two, then query three. In the diagram above, this is shown as red lines, then blue lines, then grey lines.



Option one (-db 1):

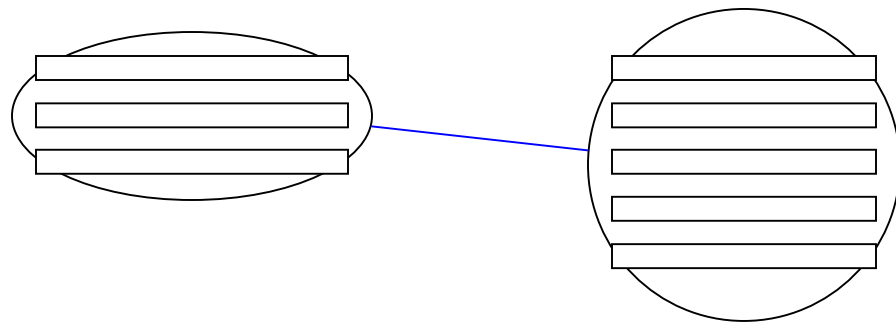
PatternHunter groups the query sequences together. It stores them all in memory and makes one pass through the subject database. Output shows the best match out of all the queries for subject sequence 1, then the next best, then the next best, etc. Then it shows the best match out of all the queries for subject sequence 2, then the next best ... and so on.

ADVANCED PATTERNHUNTER USAGE



Option two (-db 2):

PatternHunter groups the subject sequences together. It stores them all in memory and makes separate comparisons of the whole thing with each query sequence. Output shows the best match in the subject database for query sequence 1, then the next best, etc. Then it shows the best match in the subject database for query sequence 2, then the next best, and so on.



Option three (-db 3):

PatternHunter groups both subject and query sequences together and compares them all at once. Output is listed as the best two matching sequences, followed by the next best two matching sequences, and so on.

Adjusting sensitivity and speed

PatternHunter's default parameters give it an unparalleled combination of speed and accuracy. However, we may wish to adjust these settings to yield even more speed, or even more accuracy.

-w

- Applies to PHn and tPH
- Adjusts the weight of the seed used.
- Default seed has a weight of 11 for PHn and 5 for tPH
- Setting PHn's seed weight higher results in a faster search
- Setting PHn's seed weight lower results in a more sensitive search

ADVANCED PATTERNHUNTER USAGE

- PatternHunter's memory usage increases exponentially with higher seed weights (running weight 13 and above seeds is not recommended with less than 1 GB memory).

-multi

- Applies to PHn and tPH
- Adjusts the number of seeds used
- The default number of seeds is 1 for PHn and 4 for tPH
- Using more seeds increases the sensitivity of the search without sacrificing too much speed
- Using fewer seeds increases the speed of the search
- PatternHunter's memory usage is directly proportional to the number of seeds used.

-T

- Applies to tPH
- This, tolerance, represents how closely a seed must match sections of the query and subject sequence before a "hit" is triggered and the area is expanded in more detail.
- A higher tolerance results in a faster search
- A lower tolerance results in a more sensitive search
- The tolerance should be set depending on the seed weight. (For weight 5 seeds, a reasonable range is from 16-25; for weight 4, 13-20; for weight 3, 10-15)

Command List and Defaults

PatternHunter for DNA

- i Query sequence(s) [Files In]
- j Subject sequence(s) [Files In]
default = none (finds repeats within first sequence)
- o Alignment output file [File Out]
default = stdout
- db Process multiple input sequences at once (not in turn)
0=no, 1=query only, 2=subject only, 3=yes [Integer]
default = 2
- S Query sequence strands to search. 3=both, 1=top, 2=bottom [Integer]
default = 3
- a Number of processors to use [Integer]
default = 1
- mi Treat lowercase letters as masked in query sequence(s)
- mj Treat lowercase letters as masked in subject sequence(s)
- phmaski Invoke PatternHunter's internal repeat masker on query sequence(s)
- phmaskj Invoke PatternHunter's internal repeat masker on subject sequence(s)
- maskmatch Count matches among masked bases towards score
- N Number of highest scoring alignments to output per sequence pair
(max 1048576) [Integer]
default = 1048576
- Ns Number of highest scoring sequences to output [Integer]
default = 250
- O Final alignment optimization amount (eg 16) [Integer]
default = 0
- b Turn off BLAST-style alignment output
- e Maximum alignment expectation value [Real]
default = 10.0
- As Minimum alignment score [Integer]
default = 16
- r Reward for a nucleotide match [Integer]
default = 1

COMMAND LIST

- q Penalty for a nucleotide mismatch [Integer]
default = -1
- G Cost to open a gap [Integer]
default = 5
- E Cost to extend a gap [Integer]
default = 1
- W Weight of model (memory use is exponential in W; max 15) [Integer]
default = 11
- multi Range of seeds to use simultaneously (up to 16) [Integer or Range]
default = 1-1
- coding Use seed(s) optimized for DNA coding regions
- model Specify model explicitly (eg 111010010100110111) [String]
default = (optimal for model weight)
- H Number of hits needed to trigger extension, 1 or 2 [Integer]
default = 2
- h Number of hits needed to trigger local extension [Integer]
default = 3
- L Length of maximum gap [Integer]
default = 256
- s Minimum score for a HighscoringSegmentPair [Integer]
default = 16
- l Minimum length for a local HighscoringSegmentPair [Integer]
default = 16
- t Maximum distance between HSPs [Integer]
default = 64
- m Size limit on individual database partitions, in megabases [Integer]
default = 128
- P Turn off progress meter

Translated PatternHunter

- i Query sequence(s) [Files In]
- j Subject sequence(s) [Files In]
- ip Query protein sequence(s) (don't mix with -i) [Files In]
- jp Subject protein sequence(s) (don't mix with -j) [Files In]
default = none (finds repeats within first sequence)
- o Alignment output file [File Out]
default = stdout
- db Process multiple input sequences at once (not in turn)
0=no, 1=query only, 2=subject only, 3=yes [Integer]
default = 2
- S Sequence strands to search. 3=both, 1=top, 2=bottom [Integer]
default = 3

COMMAND LIST

- a Number of processors to use [Integer]
default = 1
- mi Treat lowercase letters as masked in query sequence(s)
- mj Treat lowercase letters as masked in subject sequence(s)
- M Scoring matrix file [File In]
default = BLOSUM62
- N Number of highest scoring alignments to output per sequence pair
(max 1048576) [Integer]
default = 1048576
- Ns Number of highest scoring sequences to output [Integer]
default = 250
- O Final alignment optimization amount (eg 16) [Integer]
default = 0
- b Turn off BLAST-style alignment output
- e Maximum alignment expectation value [Real]
default = 10.0
- As Minimum alignment score [Integer]
default = 50
- G Cost to open a gap [Integer]
default = 11
- E Cost to extend a gap [Integer]
default = 4
- F Cost for frameshift in a gap [Integer]
default = 10
- W Weight of model (memory use is exponential in W; max 6) [Integer]
default = 5
- multi Range of seeds to use simultaneously (up to 4) [Integer or Range]
default = 1-1
- model Specify model explicitly (eg 1101011) [String]
default = (optimal for model weight)
- T Tolerance of model hits [Integer]
default = 20
- H Number of hits needed to trigger extension, 1 or 2 [Integer]
default = 2
- L Length of maximum gap [Integer]
default = 256
- s Minimum score for a HighscoringSegmentPair [Integer]
default = 32
- l Minimum length for a local HighscoringSegmentPair [Integer]
default = 16
- t Maximum distance between HSPs [Integer]
default = 64
- P Turn off progress meter

Frequently Asked Questions

I got an OutOfMemory error, how do I get around this?

Java is preset to use a limited amount of memory: 64MB. If you get an out of memory error then you will need to tell Java to use more memory. You can tell Java to use as much memory as you have physical memory on your system. To tell java, for example, to use 512 MB of memory, type:

```
java -Xmx512m -jar PH.jar -i seq.fna
```

.

What are PatternHunter's system requirements?

PatternHunter will run on any computer that supports Java, including ordinary desktop computers. Memory requirements are dependent on the size of sequences being compared. Pentium 3 / 700 MHz or better is recommended.

My system meets PatternHunter's system requirements, but the program still runs slowly. What can I change?

Different Java Virtual Machines may change the performance speed of PatternHunter significantly. Some old Java VMs without JIT or HotSpot may reduce the run time speed by a factor of 10. Make sure that you have the most updated Java VM installed on your system.

Exactly how fast is PatternHunter when comparing two DNA sequences?

PatternHunter was used to compare the Human Genome (3 Gbase) against 16 million reads of the Mouse Genome (3 coverage, 9 Gbase total). It took only 21 days using a Pentium 3/700 MHz desktop computer with 1G of RAM. On the same machine and at the same sensitivity, Blast would take years.

How do I do a translated search?

For translated searching you must use the tPH.jar program.

Can PatternHunter query a database?

Yes. PatternHunter 2.0 supports queries against (and between) databases.

Can I see my results in BLAST-style format?

FAQ

Yes. Patternhunter now gives you the ability to see your results just as you would Blast. PatternHunter's standard output is identical to BLAST's. To use the standard PatternHunter style output, enter the -b parameter at the command prompt.

Why does the demo version give run time error like "Unable to start the application--the Java Virtual Machine cannot be loaded. Class not registered."?

Some distributions of Windows XP do not contain java VM, which PatternHunter requires. In such cases, you need to either install SP1, or download msjavx86.exe from our the BSi website at the same place as the PatternHunter demo.

How do I start PatternHunter full version from Windows?

First, make sure JDK 1.4 or better is installed on your system. Currently PatternHunter can only be started from a command line, so access the command prompt through your Start Menu -> Programs -> Accessories -> command prompt. Now type `java -Xmx512m -jar PH.jar` and you should see a list of available options.

How do I run the PatternHunter demo version?

Currently PatternHunter can only be started from a command line, so access the command prompt through your Start Menu -> Programs -> Accessories -> command prompt. From the directory containing ph.exe, type `ph`. This will tell you what options are available.

How do I compare two DNA sequences?

Suppose your sequences are stored in `seq1.fna`, `seq2.fna`. Type the command line:

```
<pre>java -jar PH.jar -i seq1.fna -j seq2.fna</pre>
```

 to compare the two sequences.

If you are running the PatternHunter demo version, replace "`java -jar PH.jar`" with "`ph`".

How do I find repeats within one DNA sequence?

Suppose your sequence is stored in a file `seq.fna`. Then type

```
<pre>java -jar PHD.jar -i seq.fna</pre>
```

 to find repeats within this sequence.

If you are running the PatternHunter demo version, replace "`java -jar PH.jar`" with "`ph`".

Can I compare a DNA sequence to itself by using it for both the 1st and 2nd sequence?

Yes, but this is not very useful because PatternHunter will find that the whole sequence matches itself and will find every repeat twice. To compare a DNA sequence to itself, you should omit the -j argument, which forces PatternHunter to ignore the trivial match and the duplicate matches.

F A Q

How do I register PatternHunter full version?

If this is your first time running PatternHunter, the program will automatically ask you to register.

About Bioinformatics Solutions Inc.

BSi provides advanced software tools for analysis of biological data.

BSI is a bioinformatics company, founded in 2000 and located at Waterloo, Ontario, Canada. Our team not only includes winners of national awards such as the Steacie Award, Killam Award, Canada Research Chairs, but also winners and gold medallists of many international competitions such as the ACM Programming Contests, the International Olympiad Informatics Contests, the Chinese National Programming and Mathematical Modeling Contest, Putnum Math Contest, Top Coder Contest, and the World Puzzle Contest. BSI has recent publications in Nature, Scientific American, Bioinformatics, and other journals.

BSI produces world's best software and algorithmic tools and solutions for the pharmaceutical industry. Some of the products include PatternHunter software for homology search, PEAKS software for peptide sequencing from MS/MS data, and PROSPECT software for predicting three-dimensional structures of proteins. BSI depends on brilliant ideas of our algorithm designers and excellent programming skills of our programmers to produce powerful software tools.

At BSI, we leverage our proprietary algorithms and novel ideas to give your business a competitive edge.

PatternHunter Software License Agreement

This is the same agreement presented on installation. It is provided here for reference only.

1. License. Subject to the terms and conditions of this Agreement, Bioinformatics Solutions (BSI) grants to you (Licensee) a non-exclusive, perpetual, non-transferable, personal license to install, execute and use one copy of PatternHunter (Software) on one single CPU at any one time. Licensee may use the Software for it's internal business purposes only.
2. Ownership. The Software is a proprietary product of BSI and is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. BSI shall at all times own all right, title and interest in and to the Software, including all intellectual property rights therein. You shall not remove any copyright notice or other proprietary or restrictive notice or legend contained or included in the Software and you shall reproduce and copy all such information on all copies made hereunder, including such copies as may be necessary for archival or backup purposes.
3. Restrictions. Licensee may not use, reproduce, transmit, modify, adapt or translate the Software, in whole or in part, to others, except as otherwise permitted by this Agreement. Licensee may not reverse engineer, decompile, disassemble, or create derivative works based on the Software. Licensee may not use the Software in any manner whatsoever with the result that access to the Software may be obtained through the Internet including, without limitation, any web page. Licensee may not rent, lease, license, transfer, assign, sell or otherwise provide access to the Software, in whole or in part, on a temporary or permanent basis, except as otherwise permitted by this Agreement. Licensee may not alter, remove or cover proprietary notices in or on the Licensed Software, or storage media; or use the Licensed Software in any unlawful manner whatsoever.

L I C E N S E A G R E E M E N T

4. *Limitation of Warranty.* THE LICENSED SOFTWARE IS PROVIDED AS IS WITHOUT ANY WARRANTIES OR CONDITIONS OF ANY KIND, INCLUDING BUT NOT LIMITED TO WARRANTIES OR CONDITIONS OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. LICENSEE ASSUMES THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE LICENSED SOFTWARE.

5. *Limitation of Liability.* IN NO EVENT WILL LICENSOR OR ITS SUPPLIERS BE LIABLE TO LICENSEE FOR ANY INDIRECT, INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER, EVEN IF THE LICENSOR OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE OR CLAIM, OR IT IS FORESEEABLE. LICENSOR'S MAXIMUM AGGREGATE LIABILITY TO LICENSEE SHALL NOT EXCEED THE AMOUNT PAID BY LICENSEE FOR THE SOFTWARE. THE LIMITATIONS OF THIS SECTION SHALL APPLY WHETHER OR NOT THE ALLEGED BREACH OR DEFAULT IS A BREACH OF A FUNDAMENTAL CONDITION OR TERM.

6. *Termination.* This Agreement is effective until terminated. This Agreement will terminate immediately without notice if you fail to comply with any provision of this Agreement. Upon termination, you must destroy all copies of the Software. Provisions 2,5,6,7 and 10 shall survive any termination of this Agreement.

7. *Export Controls.* The Software is subject at all times to all applicable export control laws and regulations in force from time to time. You agree to comply strictly with all such laws and regulations and acknowledge that you have the responsibility to obtain all necessary licenses to export, re-export, or import as may be required.

8. *Assignment.* Customer may assign Customer's rights under this Agreement to another party if the other party agrees to accept the terms of this Agreement, and Customer either transfer all copies of the Program and the Documentation, whether in printed or machine-readable form (including the original), to the other party, or Customer destroy any copies not transferred. Before such a transfer, Customer must deliver a hard copy of this Agreement to the recipient.

9. *Maintenance and Support.* BSI will provide technical support for a period of thirty (30) days from the date the Software is shipped to Licensee. Further maintenance and support is available to subscribers of BSI's Maintenance plan at BSI's then current rates. Technical support is available by phone, fax and email between the hours of 9 am and 5 pm, Eastern time, excluding statutory holidays.

10. *Governing Law.* This Agreement shall be governed by and construed in accordance with the laws in force in the Province of Ontario and the laws of Canada applicable therein, without giving effect to conflict of law provisions, and without giving effect to United Nations Convention on contracts for the International Sale of Goods.