

PFP—A Flexible Integrated Filtering and Masking Tool

Joseph A. Borkowski, Charles P. Smith and Xiaoqiu Huang

Paracel Inc., Pasadena, CA

Abstract

We have developed the **Paracel Filtering Package (PFP)**, a UNIX software package that annotates and masks repeats, and filters contaminant sequences. PFP's advantages over existing tools are its speed and flexibility, which enable it to filter and mask EST or genomic data from any organism quickly, without loss of sensitivity.

PFP is flexible in that it accepts any common input format and allows the user to specify any reference library against which to compare the input. It also offers a choice of the algorithm used for the comparison.

To quickly mask and filter large datasets such as ESTs, we have developed a hash-based software search algorithm called Haste. Optionally, the Smith-Waterman algorithm can be used in conjunction with a Paracel GeneMatcher accelerator to detect the most distantly related sequence regions for a more rigorous annotation. PFP also incorporates a version of Dust to identify low-complexity regions. When quality values are available, PFP can use this information to mask or remove low-quality regions.

PFP can use genome-sized sequences as either the reference or the query. This allows PFP to filter for contamination sequences such as *E. coli* and mitochondria in large datasets.

PFP scales linearly with the size of the input file and can handle arbitrarily large datasets, such as entire chromosomes and dbEST_human, without partitioning.

PFP provides XML support through the use of Paracel's CAML format. This format is used to store information about quality values and can mask sequence segments by adding annotation to the sequence without altering the original data.

PFP performs any number of user-defined stages in a single filtering and masking run. This allows complex filtering and masking projects to be run as a single command.

Overview of PFP Architecture

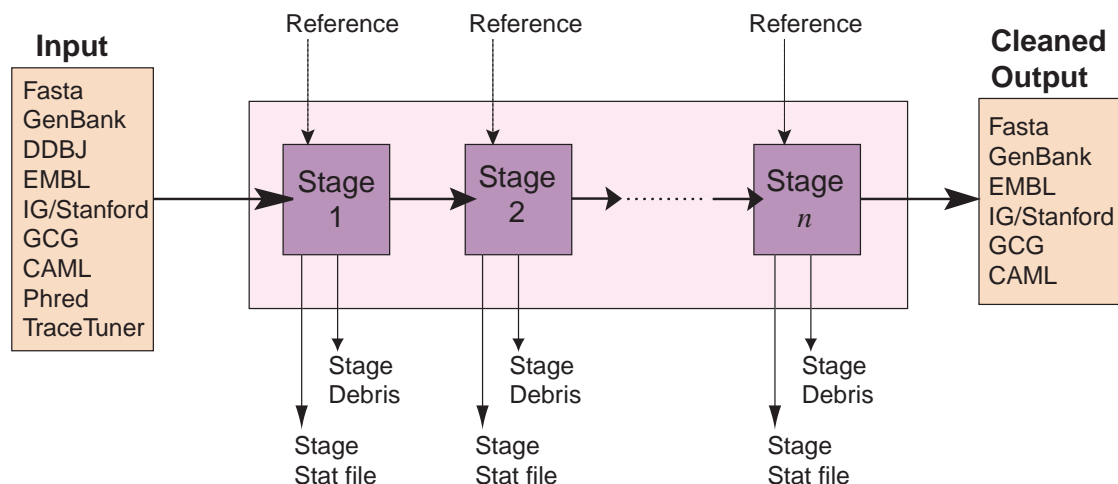


Figure 1: An overview of PFP dataflow

As input, PFP accepts a file of sequences in any of the listed formats (figure 1). This dataset is passed through any number of user-defined stages that incrementally operate on the sequences in the input file. Each stage optionally accepts a reference file containing a library of contaminant or repetitive elements. As the stage completes, it outputs statistics detailing what actions were taken and the debris (masked regions or filtered sequences) from the current stage. Each stage passes its output on to the subsequent stage; the last stage's output becomes the output of PFP itself.

The basic function of a stage is to take the stage input file, generate hits based on this file, then apply an action to the stage input file using these hits to create the stage output, debris, and statistics files (figure 2).

A stage can use one of several algorithms (see algorithm table) to generate hits based on the stage input file. To perform the comparison between the stage input file and a reference file, PFP offers a choice of a Smith-Waterman local alignment, a Needleman-Wunsch global alignment, a multiple high-scoring pairs implementation of Smith-Waterman using Paracel's GeneMatcher hardware acceleration, and the multi-threaded Haste algorithm. Developed at Paracel, Haste (Hash-Accelerated Search Tool) is a hash-based approximation of a Smith-Waterman local alignment. It is based on Xiaoqiu Huang's DDS comparison algorithm and incorporates several heuristics to improve the accuracy of the results. It belongs to the BLAST and FastA family of search algorithms, and is particularly effective for sequence screening because of its inherent ability to handle gaps and multiple high-scoring pairs.

As exhibited in the figure, this reference file is commonly a RepBase library of repetitive elements, a list of cloning vectors, or a collection of bacterial (*E. coli*), mitochondrial, or RNA contaminants.

The Dust, minlen and low-quality algorithms do not use a reference file and operate on the file itself. We have incorporated a modified and optimized variation of Dust to identify low-complexity regions. Minlen removes sequences that do not contain a given number of meaningful residues. When sequence quality values are available, a low-quality filter can be applied to remove sequence segments containing residues below a defined quality value limit.

For any of the available algorithms, the Algorithm step generates a list of hits for the stage input file. An Action step then takes these hits and uses them to perform an action on the stage input file. There are three possible actions: mask, filter, and annotate. The mask action specifies that PFP will replace any hit regions in the query with an alternative “masking” character (typically X).

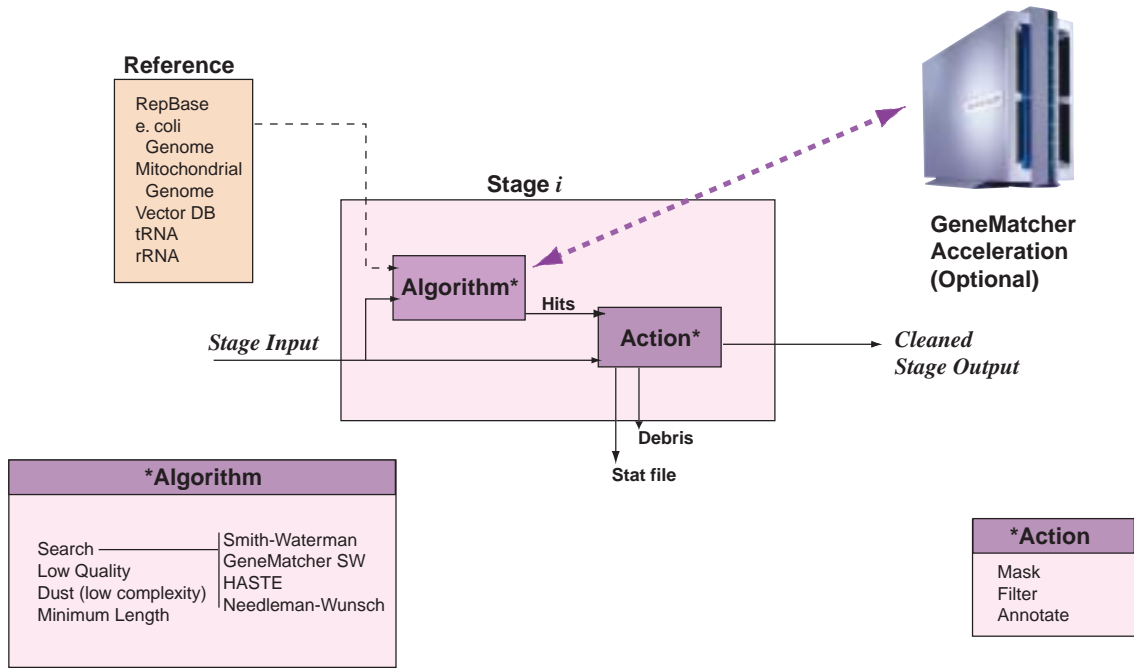


Figure 2: Expanded view of a PFP stage

The filter action removes the entire sequence containing the hit from further consideration and places it in the debris file. This is most useful for removing contaminating sequences such as *E. coli*, mitochondria, or rRNA.

The annotate action applies only if the input file type is CAML, in which case PFP will add annotations to each sequence that indicate the repeat regions that were found, without changing the underlying data. This allows repeat regions to be either used or ignored during any downstream process, e.g. clustering in the Parcel Clustering Package pipeline. In the case of clustering and assembly of ESTs, masking of repeat regions can be applied during pairwise comparison steps and removed during assembly.

The cleaned output from the action step is the output of the stage and is passed on to the next stage or as the output of PFP.

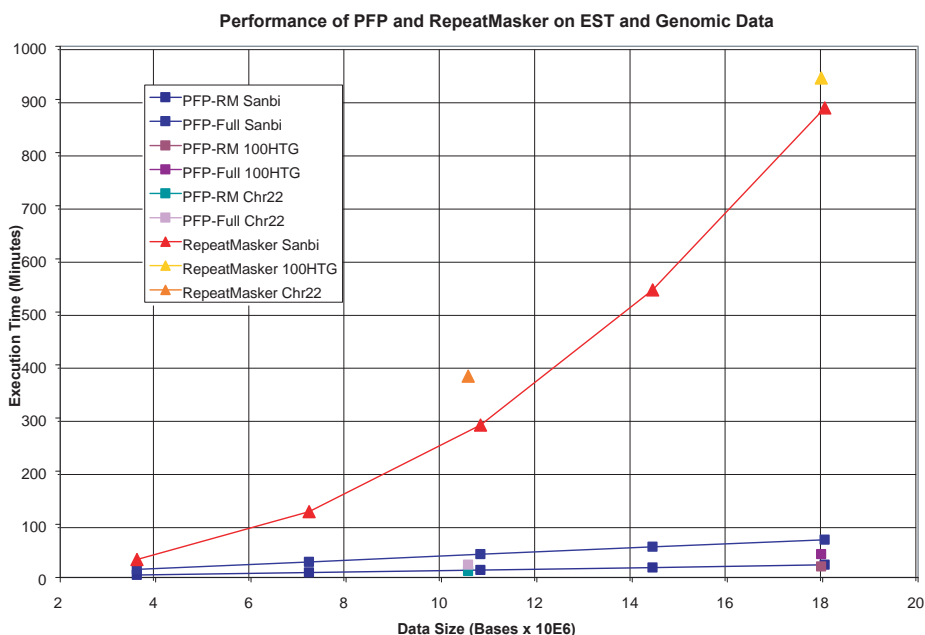
Performance

Scaling of PFP Using the Multi-Threaded Haste Algorithm

	Number of Threads			
	1	2	3	4
Human 20%	508.67 (minutes)	280.01	203.75	166.75
Sanbi 50,000	57.22	33.47	25.47	21.59

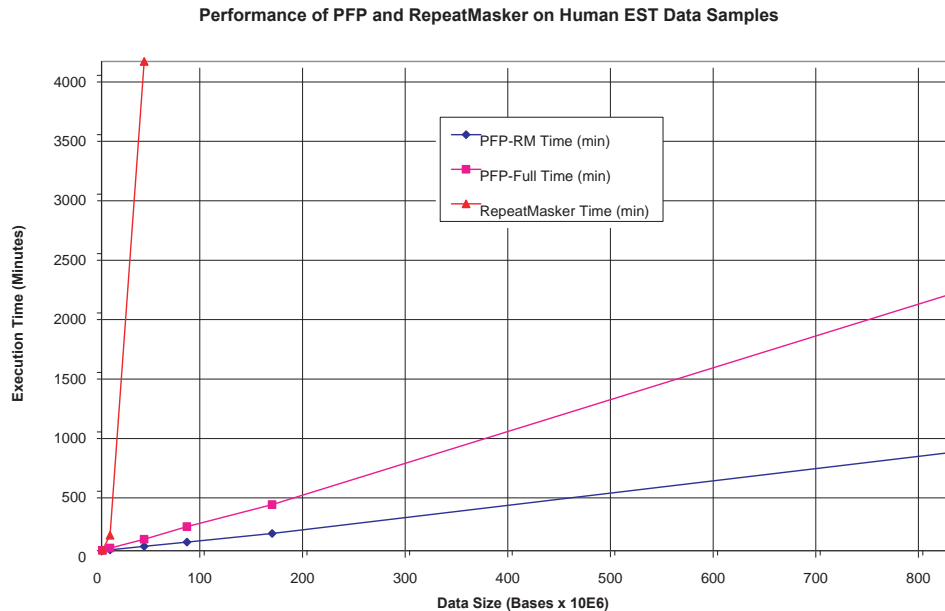
One of the features of PFP is the multi-threaded nature of the Haste algorithm, which allows the performance of PFP to scale well when additional processors are available. This table demonstrates the scaling of multi-threaded Haste on two datasets: a random 20% sample of the Human ESTs from dbEST, and 5 concatenated copies of the Sanbi 10,000 benchmark sequences. The Sanbi data are a benchmark dataset of 10,000 Human ESTs compiled by South African National Bioinformatics Institute; it is available at: <ftp://ftp.sanbi.ac.za/pub/STACK/benchmarks/benchmark10000.seq.gz>.

For each dataset, PFP was run using 1, 2, 3 or 4 processors of a quad-processor Pentium III workstation. The execution time in minutes is listed in the table. The times in this table include the execution time for Dust as a low-complexity masking stage.



This graph shows the relative performance of PFP and RepeatMasker on EST and genomic data of varying sizes. The PFP “RM” parameterization used the RepeatMasker reference files in an attempt to produce results directly comparable to RepeatMasker. The PFP “Full” runs used version 5.05 of RepBase and also searched for contaminant sequences (*E. coli*, RNA, mitochondria) in EST data. PFP “RM” and PFP “Full” used the Haste algorithm for comparisons. The PFP “GM” parameterization was the same as the “Full” run, with an additional stage that used Paracel’s GeneMatcher Smith-Waterman implementation to detect the most distantly related sequences. RepeatMasker (version 09/19/97) was run with the default settings.

There were three datasets used to generate this chart: Sanbi, 100HTG, and Chr22. The Sanbi data are the same as mentioned in the previous figure. To generate homogenous but linearly larger data for performance benchmarking, this set of 10,000 sequences was replicated to create datasets of 20,000, 30,000, 40,000, and 50,000 sequences. These five datasets were then run through PFP and RepeatMasker to generate the curves present in the graph. The Chr22 data is human chromosome 22. The 100HTG data is a sampling of 100 high-throughput genomic sequences from NCBI, averaging approximately 180,000 bases each.



This graph shows the relative performance of PFP and RepeatMasker on increasingly larger samples of human EST data that are available from dbEST. The PFP “RM” parameterization used the RepeatMasker reference files in an attempt to produce results directly comparable to RepeatMasker. The PFP “Full” runs used version 5.05 of RepBase and also searched for contaminant sequences (*E. coli*, RNA, mitochondria) in EST data. PFP “RM” and PFP “Full” used the Haste algorithm for comparisons. RepeatMaster was run with the default settings.

The full Human EST dataset used to generate these results consisted of 2,194,137 sequences for a total of approximately 830 million bases. Independent, random samples of this dataset were created at the sample rates of 0.1%, 1%, 5%, 10%, and 20%. All datasets were then run through PFP and RepeatMasker on a Pentium III workstation using a single processor with 2GB of RAM. We were not able to run the 10%, 20%, and 100% subsets as single partitions on this workstation using RepeatMasker.

Contaminant and Undesirable Sequences Found in Human EST Data Using PFP

	<i>E. coli</i>	Mitochondrial	RNA	Short Sequences
# of Sequences	865	14625	37755	121887

This table indicates the sequences that were filtered out of the Human EST dataset (2,194,137 total sequences) from dbEST in preparation for pairwise comparison and clustering. The Haste search algorithm and a high-stringency matrix (+1/-9) was utilized to identify the *E. coli*, mitochondrial, and RNA contaminants. *E. coli* hits were filtered at a score threshold of 40; mitochondrial and RNA hits were filtered at a threshold of 35. At these settings, PFP was able to identify a significant number of sequences (53245 total, or 2.4%) that matched closely with known contaminant sequence. PFP then discarded these sequences to avoid possible false-positive matches due to contamination in the pairwise comparison.

Accuracy Comparisons

Percentage of Data Masked Using PFP and RepeatMasker

	RepeatMasker Quick	RepeatMasker Default	RepeatMasker Slow	PFP RM
Human ESTs 5%	6.65% (648.29 mins)	7.28% (4,116.69)	N/A	8.33% (40.77)
Sanbi 10,000	15.99% (10.72)	17.37% (33.89)	17.63% (57.71)	18.02% (5.35)
Chr22	33.58% (226.20)	36.36% (379.17)	36.92% (628.30)	36.43% (18.30)

This table shows PFP and RepeatMasker masking different amounts of the input data and the execution time to perform that masking. Three datasets were run through PFP and RepeatMasker. The human ESTs 5% is a random sample of the human ESTs available in dbEST. The Sanbi 10,000 are the same as mentioned previously. The Chr22 data is Human chromosome 22.

PFP was run using Haste on a single processor of a Pentium III workstation using the RepeatMasker reference files in an attempt to produce results directly comparable to RepeatMasker. RepeatMasker was run in its three available levels of stringency: quick, default and slow. For each run, the percentage of input bases masked and the total execution time in minutes are reported.

Table 1: Total Execution Time (minutes)

	RepeatMasker Quick	RepeatMasker Default	RepeatMasker Slow	Censor WebSite	PFP Haste	GeneMatcher
	0.62	2.827	6.275	153.583	0.255	2.048

Table 2: Shared Masked Bases

		RepeatMasker Quick	RepeatMasker Default	RepeatMasker Slow	Censor WebSite	PFP Haste	GeneMatcher
RepeatMasker	Quick	38482					
	Default	38464	40622				
	Slow	38460	40618	41147			
Censor	WebSite	37683	39358	39698	43744		
PFP	Haste	37406	39130	39414	39770	42713	
	GeneMatcher	38158	40272	40794	41224	42181	44777

Table 3: Uniquely Masked Bases

		RepeatMasker			Censor	PFP	
		Quick	Default	Slow	WebSite	Haste	GeneMatcher
RepeatMasker	Quick	0	18	22	799	1076	324
	Default	2158	0	4	1264	1492	350
	Slow	2687	529	0	1449	1733	353
Censor	WebSite	6061	4386	4046	0	3974	2520
PFP	Haste	5307	3583	3299	2943	0	532
	GeneMatcher	6619	4505	3983	3553	2596	0

These tables show the results of an accuracy and performance comparison between PFP, RepeatMasker and Censor. The data are a small set of three high-throughput genomic sequences (141,151 bases total) downloaded from NCBI. To generate the Censor results, the dataset was submitted to the Censor web site (<http://charon.girinst.org/~server/censor.html>). RepeatMasker was run in its three available levels of stringency: quick, default and slow. PFP was run using the same reference files and matrices as RepeatMasker. Two PFP runs were made to find the repetitive regions: one using Haste and the other using Smith-Waterman accelerated by Paracel's GeneMatcher.

Table 1 indicates the total execution time in minutes for running this dataset through the programs.

To evaluate the relative accuracy of these masking methods we examined the output of each of the runs to see which regions of the query were masked. For any two masking methods, we were able to compare the two outputs and count the number of bases that both the methods masked ("shared masked bases"). These data are presented in Table 2. A higher number for shared masked bases indicates that the two methods masked more of the same regions. We also counted the number of uniquely masked bases—bases that one method masked and the other did not. These data are presented in Table 3. The numbers present in the table are the number of unique bases found by the method of that row (name in left-hand column) when compared against the method of that column. For example, RepeatMasker-Quick masked 18 bases that RepeatMasker-Default did not mask; RepeatMasker-Default masked 2,158 bases that RepeatMasker-Quick did not mask.

Conclusion

- ▶ PFP is fast and scales well on large datasets
 - ▶ PFP is accurate compared to other masking packages
 - ▶ PFP is flexible and can filter and mask complex datasets such as dbest and genomic sequences
-

Acknowledgments:

We would like to acknowledge the assistance of Cassi Paul, Matt Mettler, Tim Hunkapiller, Gene Shpaer, James Candlin, Licen Xu, Glen Herrmannsfeldt, Stephanie Pao, Tristan Gill and Joe Salvatore.

PARACEL[®]
A CELERA BUSINESS

80 South Lake Avenue, Suite 650
Pasadena, CA 91101-2693
Phone: 626.744.2000 • Fax: 626.744.2001
www.paracel.com