

pfsearch next generation: accelerated search of PROSITE profiles

Thierry Schuepbach^{1,3}, Marco Pagni¹, Alan Bridge², Lydie Bougueleret², Ioannis Xenarios^{1,2}, Lorenzo Cerutti²

¹Vital-IT group, SIB Swiss Institute of Bioinformatics, Genopode, UNIL-Sorge, CH-1015 Lausanne

²Swiss-Prot group, SIB Swiss Institute of Bioinformatics, CMU, Rue Michel-Servet 1, CH-1211 Geneva

³Molecular Modeling group, SIB Swiss Institute of Bioinformatics, Genopode, UNIL-Sorge, CH-1015 Lausanne

Summary

The PROSITE (<http://prosite.expasy.org>) resource provides a rich and well annotated source of signatures in the form of generalized profiles that allow protein domain detection and functional annotation. One of the major limiting factors in the application of PROSITE in genome and metagenome annotation pipelines is the time required to search protein sequence databases for putative matches. We describe an improved and optimized implementation of the PROSITE search tool *pfsearch* that is intended to address this limitation.

An heuristic for pfsearch

A major reduction in the execution time of sequence database searches can be achieved by introducing an heuristic pre-filter that selects candidate sequences. We implemented an heuristic for generalized profiles, named *prfh*:

- ▶ each position i of the profile and j of the sequence we define a score $S(i, j)$:

$$S(i, j) = \max \begin{cases} S(i-1, j-1) + M(i, a_j) \\ 0 \end{cases}$$

where $M(i, a_j)$ is the match score read at position i of the profile matrix table for residue a_j observed at position j of the sequence.

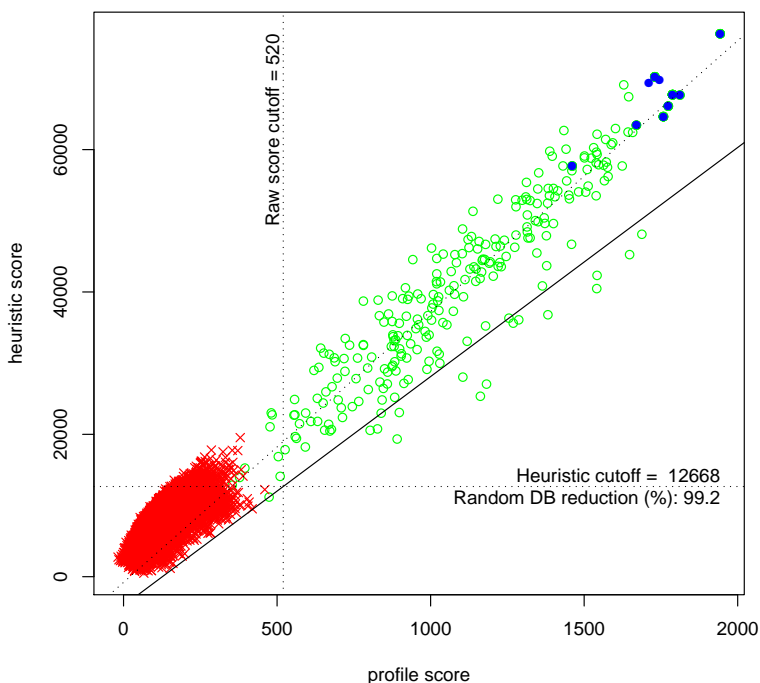
- ▶ only the maximal scoring diagonal $S(i, j)$ is kept for every position j of the sequence. All maxima are then summed to form the final heuristic score (H_{score}).

$$H_{score} = \sum_j (\max_i S(i, j))$$

Calibration of the heuristic for PROSITE profiles

The H_{score} distribution linearly correlates with the raw score distribution obtained using the standard *pfsearch* ($R^2 \approx 0.9$ on average). We use this property to determine the appropriate H_{score} cutoffs with respect to the normalized score cutoffs of each calibrated profile:

- ▶ randomly sample 200 sequences belonging to the original seed alignment (re-sampling if their number is <200);
- ▶ generate a set of artificially mutated sequences based on the sampled sequences, including indels, at various PAM distances sharing from 40% to 85% sequence identity with their source;
- ▶ score the sequences with both the standard profile scoring method and the heuristic;
- ▶ calculate the regression line on the lower 5% quantile of the heuristic score distribution using the quanteg R package, and use it to obtain the heuristic cutoffs corresponding to the standard profile cutoffs. (NB: The regression on a low quantile ensures a minimal loss of true positive sequences)



Example: estimation of the H_{score} cutoff from the regression line of the lower 5% quantile (black line) for profile CYTOCHROME_B5_2 ($R^2=0.9$):

- sequences from seed alignment;
- × shuffled sequences;
- simulated sequences.

The new H_score keyword

We introduced a new keyword *H_score* to in PROSITE profiles to inform the program of the heuristic cutoffs. The keyword do not interfere with the previous version of the *pfsearch* program.

```
ID  AMMECR1; MATRIX.
AC  PS51112;
DT  APR-2005 (CREATED); APR-2005 (DATA UPDATE); JAN-2012 (INFO UPDATE).
DE  AMMECR1 domain profile.
MA  /GENERAL_SPEC: ALPHABET='ABCDEFGHIKLMNPQRSTVWYZ'; LENGTH=194;
MA  /DISJOINT: DEFINITION=PROTECT; N1=6; N2=189;
MA  /NORMALIZATION: MODE=1; FUNCTION=LINEAR; R1=1.0061111; R2=0.0089461; TEXT='-LogE';
MA  /CUT_OFF: LEVEL=0; SCORE=838; N_SCORE=8.5; H_SCORE=43007; MODE=1; TEXT='!';
MA  /CUT_OFF: LEVEL=-1; SCORE=615; N_SCORE=6.5; H_SCORE=32704; MODE=1; TEXT='?';
.
```

Software optimization

Pfsearch has been rewritten and optimized in C from the original Fortran:

- ▶ we entirely reformatted the memory structure to allow vectorization;
- ▶ high level assembly code (intrinsic functions) was used to enforce the SSE 2 and SSE 4.1 instruction sets;
- ▶ added multi-thread support.

On a single core, the new software implementation runs $2\times$ faster than the original Fortran. This acceleration scales up with multi-threading: on a dual hyperthreaded quad-core machine we measured an average 10-fold improvement. The sequence database was also indexed to avoid repeated disk access for sequence scanning, particularly useful for repeated calls when searching multiple profiles.

Performance

The performance obtained by combining the developed heuristic and the software optimization has been measured for the 904 PROSITE profiles, for which we were able to fix the *H_score* cutoff, against 16,544,936 UniProtKB sequences (5,358,014,649 residues). All measures have been obtained using a dual hyperthreaded quad-core Intel workstation.

	Mean	Median
TP recovery	99.6%	99.9%
Database reduction (heuristic)	96.7%	99.1%
Search time/profile:		
pfsearchV3	98 sec	73 sec
pfsearch on Vital-IT cluster	5-10 mins ^(*)	-
standard pfsearch	3 hours ^(*)	-

^(*) estimated time for a profile of length 120.

NB: most of the missing TP matches correspond to fragment sequences.

Conclusion

By combining the heuristic with our code optimization we achieved a 100x fold increase in the speed of *pfsearch* on average on a modern workstation. For example the human proteome can be searched with the totality of the PROSITE profile models in less than 4 hours and this time can be drastically reduced on machines with a large number of CPU cores and/or computer clusters.

