Acceleration of the pfsearch program to search PROSITE generalized profiles

Thierry Schuepbach, Marco Pagni, Alan Bridge, Lydie Bougueleret , Ioannis Xenarios, Lorenzo Cerutti

Introduction

Functional annotation of new proteins is one of the most important tasks in the process of describing a new sequenced organism [*e.g.* in Wurm et al., (2011)]. Sequence homology methods are commonly used to infer the function of a protein from well annotated templates. Among sequence homology methods, profile techniques have been widely and successfully used to annotate new sequences since they can detect more subtle homologies than standard pairwise techniques and usually produce better alignments to the template sequences, allowing the transfer of annotation at both the protein and the residue level.

PROSITE generalized profiles combined with rules and patterns are very efficient for domain detection and functional annotation (Sigrist *et al.*, 2010) thanks to both the quality of annotation attached to the models that can be transferred to new sequences and the quality of the manually built profile models.

Profile methods need to be fast to keep pace with the daily increase of sequences produced using the new sequencing techniques and many efforts focus in the development of fast and efficient code (Eddy, 2011; Remmert et al., 2011). In this manuscript we describe the implementation of a simple heuristic and the code optimization and parallelization of the *pfsearch* program used to search sequence databases with generalized profiles. On a modern x86 computer (dual hyperthreaded quad-core) we measure an increase of speed of 2 orders of magnitude higher than the original search algorithm without losing the benefits and flexibility of the PROSITE generalized profiles.

Heuristics and calibration

A major reduction of the execution time to search a sequence database with a profile can be achieved by introducing an heuristic-filter step to rapidly select candidates to pass to the more CPU-expensive search and alignment core algorithms, similarly to what has be done for PSI-BLAST (Altschul *et al.*, 1997) and HMMER3 (Eddy, 2011). Although the heuristic step doesn't guarantee to find the maximum scoring regions, thus some true positive matches may be lost, the gain in speed is essential to deal with the accelerating accumulation of new sequences via next generation sequencing technologies.

We implemented a simple heuristic, named *prfh*, to score maximal matching diagonals between the profile and the sequence without considering gap penalties neither the order of the match diagonals. For each position *i* of the profile and for each position *j* of the sequence we define a score *S(i,j)*:

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

where *P(i,a_j)* is the score read at position *i* of the profile matrix table for residue a_j observed at position *j* of the sequence. Boundary scores *S(i,0)* and *S(0,j)* are set to 0. Equation (1) is similar to the standard Smith-Waterman algorithm (Smith and Waterman, 1981) without considering gaps. Successively only the maximal scoring diagonal *S(i,j)* is kept for a given position *j* of the sequence [the maximization part of equation (2)]. Finally, all the maxima are summed to form the final heuristic score

$$H_{score} = \sum_j (\max_i S(i,j)) \qquad (2)$$

Note that given equation (2) we only need to store the single row vector *S(i-1,•)* of matrix *S(i,j)* and update it directly as we proceed on the sequence position *j* of equation (2), which can be easily vectorized using modern CPU instructions.

For most of the PROSITE profiles, the H_{score} distribution linearly correlates with the score distribution obtained using the standard *pfsearch* scoring algorithm (average coefficient of determination $R^2 = 0.9$). By determining the parameters of this correlation we can fix the heuristic cutoffs relatively to the standard score cutoffs read from the calibrated PROSITE profile (Sigrist *et al.*, 2002). Thus, we can directly fix the heuristic score cutoff based on the type of search we want to perform: strict, with a standard profile normalized score cutoff usually set at 8.5, or more exploratory with usually a normalized cutoff score set at 6.5 (Pagni and Jongeneel, 2001).

To precisely estimate the linear correlation parameters between the heuristic and the standard profile scores we need sequences producing all possible range of scores, from high scoring sequences to the poor scoring ones. Unfortunately at the time of the profile construction only the sequences participating in the seed alignment are known, which by definition are those producing high scores. To overcome this limitation we randomly selected 200 sequences belonging to the original seed alignment for each profile (sequences are re-sampled if their number is inferior to 200). New sequences are generated by artificial mutation of the original selected sequences with various PAM distances (20PAM, 40PAM, 60PAM, 80PAM, 100PAM) and the final 1000 sequences (with identities to the original sequences ranging from ~85% to ~40%) are scored independently with both the standard profile scoring method and the heuristic. A regression line can be calculated and used to map the heuristic cutoffs given the profile cutoffs. We chose to calculate the regression line based on the first 5% quantile of the heuristic score distribution. This has the effect to lower the regression line of the heuristic *vs* the standard profile scores, resulting in a lowering of the heuristic cutoffs, thus ensuring a minimal loss of true positives at the price to recover more false positives.

The linear regressions permitts to fix automatically the heuristic-filter cutoffs in ~3/4 of the PROSITE profiles. For the remaining profiles we made some adjustments manually. This is mostly due to special profiles, *e.g.* profiles with special cutoffs levels, or due to our "pessimistic" cutoffs based on the 5% quantile regression line which in some cases resulted in a poor performance of the heuristic filter by recovering too many false positives. A small number of profiles cannot be used with heuristic: circular profiles, because of their special topology; very short profiles, which are not properly detected by the heuristic; and a small number of profiles which need to be rebuild to correct some score anomalies resulting from translations from HMMs without proper rescaling.

Software optimization and performance of the new pfsearch

To exploit the new capabilities of modern multicore processors, the core algorithms of *pfsearch* have been rewritten and optimized in C from the original Fortran code: *xali1*, responsible to pre-filter the database; *xalip*, responsible to detect and score the matches; *xalit* responsible for the construction of the final alignment. The optimization process, which also includes the heuristic-filter algorithm, entirely reformatted the memory structure to allow vectorization and high level assembly code (intrinsic functions) has been used to enforce modern processors instruction set (SSE 4.1 and SSE 4.2), which leads to an acceleration of 2x than the original Fortran code. We also introduced an index of the sequence database to avoid repeated sequence scanning, particularly useful for repeated calls when searching multiple profiles, resulting in a gain of 50sec per search in average of the final search time.

The obtained acceleration scales up with multithreading, *e.g.* on a dual hyperthreaded quad-core machine we measured an average improvement of 10x in speed to search a profile *vs* a protein database. By adding the heuristic to pre-filter the sequences in the database we measured a supplementary decrease in execution time of 10x in average, resulting in a 100x speed up in average. This increase in performance is comparable to other modern profile database search algorithms. To search 16,544,936 sequences (5,358,014,649 residues) from UniProtKB, we measured a mean time of 98 sec/profile (median of 73 sec/profile). For 99% of the PROSITE profiles for which we where able to fix an heuristic cutoff, the heuristic-filter was able to recover ≥ 98% of true positive matches (92% of them have a recovery of ≥ 99%, worst recovery measured is 92.6%), which was the loss expected by the introduction of an heuristic algorithm. To recover the totality of the true positive sequence the heuristic-filter can be inactivated by the user, therefore the search speed will depends only on the number of CPU cores available.

Availability

The source code and binaries of the new *pfsearch* are available from the ExPASy website: http://web.expasy.org/pftools. PROSITE generalized profiles including the heuristic cutoff are also available at the same address. These profiles will be soon integrated in the official PROSITE release.

The regression method to fix the heuristic cutoff uses Perl and R scripts. These are available upon request, and soon replaced be a standalone program which will be distributed with the new *pfsearch* code.

Acknowledgments

Persons: Philip Bucher, Denis,Beatrice,Christian, other?

Grants …

References

Altschul,S.F. *et al.* (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research*, **25**, 3389–402.

Eddy,S.R. (2011) Accelerated Profile HMM Searches. *PLoS Computational Biology*, **7**, e1002195.

Pagni,M and Jongeneel,C.V. (2001) Making sense of score statistics for sequence alignments. *Briefings in bioinformatics*, **2**, 51–67.

Remmert,M. *et al.* (2011) HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. *Nature Methods*, **9**, 173–5.

Sigrist,C.J.A. *et al.* (2010) PROSITE, a protein domain database for functional characterization and annotation. *Nucleic acids research*, **38**, D161–6.

Sigrist,C.J.A. *et al.* (2002) PROSITE: a documented database using patterns and profiles as motif descriptors. *Briefings in bioinformatics*, **3**, 265–74.

Smith,T.F. and Waterman,M.S. (1981) Identification of common molecular subsequences. *Journal of molecular biology*, **147**, 195–7.

Wurm,Y. *et al.* (2011) The genome of the fire ant Solenopsis invicta. *Proceedings of the National Academy of Sciences of the United States of America*, **108**, 5679–84.