Bioinformatics, 31(23), 2015, 3841–3843
doi: 10.1093/bioinformatics/btv451
Advance Access Publication Date: 30 July 2015
Applications Note

Sequence analysis

AlignBucket: a tool to speed up 'all-against-all' protein sequence alignments optimizing length constraints

Giuseppe Profiti[1,2,3], Piero Fariselli[1,2,*] and Rita Casadio[2,3]

[1]Department of Computer Science and Engineering, via Mura Anteo Zamboni 7, Bologna, [2]Bologna Biocomputing group, via S. Giacomo 9/2, Bologna and [3]Health Sciences and Technologies ICIR, via Tolara di Sopra 41/E, Ozzano dell'Emilia, Italy

*To whom correspondence should be addressed.

Associate Editor: Burkhard Rost

Received on April 14, 2015; revised on July 3, 2015; accepted on July 24, 2015

Abstract

Motivation: The next-generation sequencing era requires reliable, fast and efficient approaches for the accurate annotation of the ever-increasing number of biological sequences and their variations. Transfer of annotation upon similarity search is a standard approach. The procedure of all-against-all protein comparison is a preliminary step of different available methods that annotate sequences based on information already present in databases. Given the actual volume of sequences, methods are necessary to pre-process data to reduce the time of sequence comparison.

Results: We present an algorithm that optimizes the partition of a large volume of sequences (the whole database) into sets where sequence length values (in residues) are constrained depending on a bounded minimal and expected alignment coverage. The idea is to optimally group protein sequences according to their length, and then computing the all-against-all sequence alignments among sequences that fall in a selected length range. We describe a mathematically optimal solution and we show that our method leads to a 5-fold speed-up in real world cases.

Availability and implementation: The software is available for downloading at http://www.biocomp.unibo.it/~giuseppe/partitioning.html.

Contact: giuseppe.profiti2@unibo.it

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

'All-against-all' sequence alignments in protein datasets are routinely performed with programs such as BLAST (Altschul *et al.*, 1990) or dynamic programming approaches (Needleman and Wunsch, 1970). By this, sequences group into subsets of sequence identity values that are important in several applications, including transfer of structural and functional features between protein sequences (for details see www.uniprot.org/program/ and www.ebi.ac.uk/GOA/ElectronicAnnotationMethods). Proteins that share at least some 30% of sequence identity may overlap in their three-dimensional structures (Chothia and Lesk, 1986), while proteins sharing at least some 60% of sequence identity are likely to share also similar functions (Rost, 2002; Tian and Skolnick, 2003). It is also important to consider the relative length of the pairwise aligned sequences before transferring annotations (Devos and Valencia, 2001). Multidomain proteins are less functionally conserved than single-domain ones (Hegyi and Gerstein, 2001) and methods based on transfer of function after homology search can group proteins with different functions in the same cluster. Therefore, both sequence identity values and the percentage of overlapping positions over the alignment length (coverage) were constrained in developing methods for annotation problems (Miele *et al.*, 2011; Piovesan *et al.*, 2011, 2013). Evolutionary studies include alignments in the early stage of their workflow (Vilella *et al.*, 2009; Waterhouse *et al.*,

Table 1. Optimal partitioning into sets as a function of a minimal bound coverage of the sequence alignment

δ	Sets (no.)	Cost for sets	% of full cost	Speedup
0.0	1	3.85×10^{17}	100.00	1.00
0.1	2	3.85×10^{17}	99.99	1.00
0.2	3	3.85×10^{17}	99.85	1.00
0.3	4	3.81×10^{17}	98.97	1.01
0.4	5	3.56×10^{17}	92.30	1.08
0.5	8	3.12×10^{17}	80.95	1.24
0.6	11	2.53×10^{17}	65.66	1.52
0.7	18	1.85×10^{17}	47.95	2.09
0.8	30	1.19×10^{17}	30.82	3.25
0.9	65	5.62×10^{16}	14.58	6.86

Database is UniprotKB 2015_05 with a total of about 32 million sequences; δ is the minimal expected coverage required in the pairwise sequence alignment. Cost: sum of Equation (1) over the sets. Full cost is for $\delta = 0$. Speed up is full cost over cost (Supplementary Materials).

2013), as well as different computational methods to speed up computation (Roth *et al.*, 2008; Sonnhammer *et al.*, 2014; Wittwer *et al.*, 2014).

Considering the computational cost, the all-against-all similarity search is an extremely demanding process. Recently, in problems related to evolutionary studies, Wittwer *et al.* (2014) showed that by considering subsequence-level homology, the speed of the all-against-all alignment process could be increased by a factor of four, while maintaining the possibility of retrieving homologous sequences. On the same line of research, here we present an algorithm that can speed up the computational time of the similarity search by constraining the expected alignment coverage.

2 Methods

According to Altschul *et al.* (1990), the BLAST computational cost (c) of comparing all pairs of sequences with lengths ranging from s to t is:

$$c(s,t) = \sum_{k=s}^{t} S_k \cdot (k + \sum_{j=s}^{t} jS_j) \quad (1)$$

where k and j are sequence lengths (number of residues) and S_k is the number of sequences of length k in the database. Given a bound (δ) on the required minimum coverage (fraction of overlapping positions over the sequence alignment length, ≤ 1) of the resulting alignment, all the sequences in the database can be partitioned in different ways. The two extreme possibilities are (i) a single set containing all the sequences, and the procedure of the all-against-all pairwise sequence comparison performs many worthless computations; (ii) N sets (one for each possible sequence length k), including all the sequences in the range $[\lceil k \times \delta \rceil, \lfloor k \div \delta \rfloor]$, and the procedure of the all-against-all pairwise sequence comparison performs again many useless alignments. In case (i) many alignments will not meet the required bound δ, while in case (ii) too many duplicated comparisons will be performed.

To reduce the alignment time, we introduce AlignBucket, a procedure that optimizes sequence-length bounds for each subset of the database to minimize the computational cost of Equation (1) (for theoretical details see Supplementary Materials).

3 Results and conclusion

The algorithm was tested on UniprotKB (UniProt, 2011, release 2015_05), including 32 688 435 sequences. The dataset contains unique protein sequences with length ranging from 40 to 36, 805 residues (the length distribution is in Supplementary Fig. 1S). According to Equation (1), an all-versus-all sequence comparison of the dataset leads to a total cost of 3.85×10^{17}, when δ is set to 0 (Table 1).

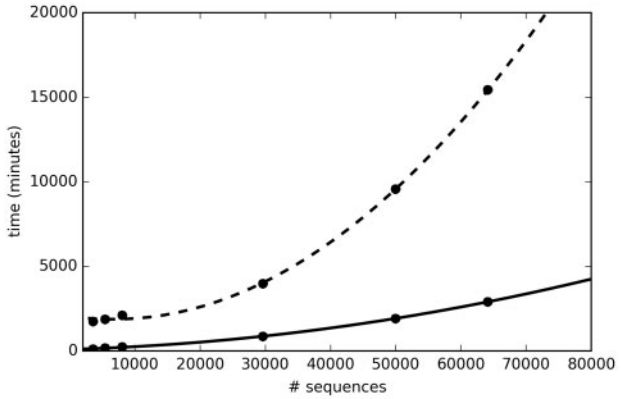

Fig. 1. Comparison of Blast running times with and without AlignBucket at increasing number of sequences. Reference database and sequences are from SwissProt 2015_05. Running times are in minutes on a single computational core. Dashed line: Blast performs an all-versus-all pairwise sequence comparison. Solid line: Blast compares the same sequences partitioned into 77 subsets identified by AlignBucket. Length distribution of the sample is the same as in the original database. The pairwise coverage value δ is 0.9 (Supplementary Materials)

At increasing δ values from 0.1 to 0.9, the computational cost for the increasing number of sets defined with AlignBucket decreases up to 6.86 times. Table 1 lists the optimal number of sets, the evaluated cost, its percentage value and the computational speedup of the selected partition over a single set containing all the sequences.

When the algorithm is tested on real world data, a speed-up of 5.3 is already measured when 65 000 sequences are compared with 548 458 chains of the SwissProt database (Fig. 1). This result is consistent with the theoretical estimate (Table 1).

In conclusion, here we describe a new algorithm for partitioning protein sequences given a constraint on the expected alignment coverage and our contribution points to the pre-processing algorithm that filters out unnecessary comparisons. Alignments in the buckets are computed as those in the all-versus-all comparisons, and therefore, the only missing outputs are the alignments between pairs of sequences that would be discarded by a coverage filter in post-processing. The partition found by the algorithm can be proven to be optimal (for details refer to Supplementary Materials).

Funding

This work was supported by COST BMBS Action TD1101 and Action BM1405 (EU RTD Framework Program, to R.C.); PON projects PON01_02249 and PAN Lab PONa3_00166 Italian Ministry of University and Research, (to R.C.) and FARB-UNIBO 2012 (to R.C.).

Conflict of Interest: none declared.

References

Altschul,S.F. *et al.* (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Chothia,C. and Lesk,A.M. (1986) The relation between the divergence of sequence and structure in proteins. *EMBO J.*, **5**, 823.

Devos,D. and Valencia,A. (2001) Intrinsic errors in genome annotation. *Trends Genet.*, **17**, 429–431.

Hegyi,H. and Gerstein,M. (2001) Annotation transfer for genomics: measuring functional divergence in multi-domain proteins. *Genome Res.*, **11**, 1632–1640.

Miele,V. *et al.* (2011) Ultra-fast sequence clustering from similarity networks with silix. *BMC Bioinformatics*, **12**, 116.

Needleman,S.B. and Wunsch,C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, **48**, 443–453.

Piovesan,D. *et al.* (2011) Bar-plus: the bologna annotation resource plus for functional and structural annotation of protein sequences. *Nucleic Acids Res.*, **39**(Suppl 2), W197–W202.

Piovesan,D. *et al.* (2013) How to inherit statistically validated annotation within BAR+ protein clusters. *BMC Bioinformatics*, **14**(Suppl 3), S4.

Rost,B. (2002) Enzyme function less conserved than anticipated. *J. Mol. Biol.*, **318**, 595–608.

Roth,A.C. *et al.* (2008) Algorithm of oma for large-scale orthology inference. *BMC Bioinformatics*, **9**, 518.

Sonnhammer,E.L. *et al.* (2014) Big data and other challenges in the quest for orthologs. *Bioinformatics*, **30**, 2993–2998.

Tian,W. and Skolnick,J. (2003) How well is enzyme function conserved as a function of pairwise sequence identity?. *J. Mol. Biol.*, **333**, 863–882.

UniProt (2011) Ongoing and future developments at the universal protein resource. *Nucleic Acids Res.*, **39**(Suppl 1), D214–D219.

Vilella,A.J. *et al.* (2009) Ensemblcompara genetrees: complete, duplication-aware phylogenetic trees in vertebrates. *Genome Res.*, **19**, 327–335.

Waterhouse,R.M. *et al.* (2013) Orthodb: a hierarchical catalog of animal, fungal and bacterial orthologs. *Nucleic Acids Res.*, **41**, D358–D365.

Wittwer,L.D. *et al.* (2014) Speeding up all-against-all protein comparisons while maintaining sensitivity by considering subsequence-level homology. *PeerJ*, **2**, e607.