Speeding up all-against-all protein comparisons while maintaining sensitivity by considering subsequence-level homology
journal article
en

Speeding up all-against-all protein comparisons while maintaining sensitivity by considering subsequence-level homology

Orthology inference and other sequence analyses across multiple genomes typically start by performing exhaustive pairwise sequence comparisons, a process referred to as "all-against-all". As this process scales quadratically in terms of the number of sequences analysed, this step can become a bottleneck, thus limiting the number of genomes that can be simultaneously analysed. Here, we explored ways of speeding-up the all-against-all step while maintaining its sensitivity. By exploiting the transitivity of homology and, crucially, ensuring that homology is defined in terms of consistent protein subsequences, our proof-of-concept resulted in a 4x speedup while recovering >99.6% of all homologs identified by the full all-against-all procedure on empirical sequences sets. In comparison, state-of-the-art k-mer approaches are orders of magnitude faster but only recover 3-14% of all homologous pairs. We also outline ideas to further improve the speed and recall of the new approach. An open source implementation is provided as part of the OMA standalone software at http://omabrowser.org/standalone.

Speeding up all-against-all protein comparisons while maintaining sensitivity by considering subsequence-level homology

Lucas D. Wittwer[1,2,3,#], Ivana Piližota[1,#], Adrian M. Altenhoff[1,2,3], Christophe Dessimoz[1,2,*]

[1]University College London, Gower St, London WC1E 6BT, United Kingdom
[2]Swiss Institute of Bioinformatics, Universitätstr. 6, 8092 Zurich, Switzerland
[3]ETH Zurich, Computer Science, Universitätstr. 6, 8092 Zurich, Switzerland

#These authors contributed equally to this work.
*Corresponding author. Email: c.dessimoz@ucl.ac.uk

Abstract

Orthology inference and other sequence analyses across multiple genomes typically start by performing exhaustive pairwise sequence comparisons, a process referred to as "all-against-all". As this process scales quadratically in terms of the number of sequences analysed, this step can become a bottleneck, thus limiting the number of genomes that can be simultaneously analysed. Here, we explored ways of speeding-up the all-against-all step while maintaining its sensitivity. By exploiting the transitivity of homology and, crucially, ensuring that homology is defined in terms of consistent protein subsequences, our proof-of-concept resulted in a 4x speedup while recovering >99.6% of all homologs identified by the full all-against-all procedure on empirical sequences sets. In comparison, state-of-the-art *k*-mer approaches are orders of magnitude faster but only recover 3-14% of all homologous pairs. We also outline ideas to further improve the speed and recall of the new approach. An open source implementation is provided as part of the OMA standalone software at http://omabrowser.org/standalone.

1 Introduction

Advances in genome sequencing have led to an immense increase in the number of available genomes (Metzker 2009; Pagani et al. 2012). As the experimental annotation of these sequences would be prohibitively slow and expensive, there is a strong interest in computational methods (reviewed in Rentzsch and Orengo 2009).

33 Most approaches start by identifying genes of common ancestry within and across
34 species—sets of "homologous" genes (Patterson 1988). As further refinement,
35 homologous proteins, which can be split up into paralogs and orthologs, diverged
36 from a common ancestral protein (Fitch 1970). Paralogous sequences, which start
37 diverging through gene duplication, are believed to drive function innovation and
38 specialisation, whereas orthologous sequences, which diverged through speciation,
39 tend to have more similar biological function (Tatusov 1997; Lynch 2000; Altenhoff et
40 al. 2012; Gabaldón and Koonin 2013). Therefore, protein function can be assigned by
41 identifying regions of orthology between a newly sequenced and an already
42 annotated genome (e.g., Gaudet et al. 2011). Orthology inference is also of high
43 interest to infer phylogenetic species trees and to study the evolution of gene
44 families (Altenhoff and Dessimoz 2012).

45 Due to the high biological interest for orthology detection, many orthology inference
46 algorithms and associated databases have been developed (reviewed in Dessimoz
47 et al. 2012; Sonnhammer et al. 2014). Some of these can be large, with databases
48 comparing thousands of genomes of different species: eggNOG with currently 3686
49 genomes (Powell et al. 2013), Roundup with 2044 (DeLuca et al. 2012), OMA with
50 1613 genomes (Altenhoff et al. 2011) and OrthoDB with 1425 genomes (Waterhouse
51 et al. 2013).

52 To detect homology, these resources rely on all-against-all protein comparison. All
53 pairs of sequences are aligned using BLAST (Altschul et al. 1997) or dynamic
54 programming (Smith and Waterman 1981). They are then inferred as homologous if
55 their alignment score is above a certain threshold and alignment length constraints
56 are satisfied (Dessimoz et al. 2005; Roth et al. 2008). Even orthology databases
57 based on reconciled gene/species trees (Vilella et al. 2008) or hidden Markov model
58 profiles (Mi et al. 2013) typically start their procedure with an all-against-all step,
59 before proceeding to clustering.

60 However, the all-against-all procedure scales quadratically in the number of
61 sequences compared and hence rapidly becomes very costly. For instance, the all-
62 against-all phase in OMA, the resource developed in our lab, has already consumed
63 over 6.1 million CPU hours to date, despite the use of a highly optimised
64 implementation (Szalkowski et al. 2008) of the Smith-Waterman (1981) pairwise
65 alignment algorithm. Furthermore, in the all-against-all phase of OMA, about half of

66 the time is spent on unrelated gene pairs (see Supplementary materials) which are

67 not used for further orthology inference.

68 Previous methods have been proposed to speedup the all-against-all phase, but they

69 tend to come at a cost of lower sensitivity. CD-HIT (Li and Godzik 2006), UCLUST

70 (Edgar 2010), and kClust (Hauser et al. 2013) are methods to reduce the complexity

71 by clustering the dataset based on the number of common or high-scoring words (k-

72 mers) in each sequence. This works well to identify closely related sequences—e.g.,

73 close paralogs or "redundant" sequences—but such approaches are less sensitive

74 than full dynamic programming alignment and thus tend to miss distant homologs.

75 Another approach was proposed in the Hieranoid method (Schreiber and

76 Sonnhammer 2013): by using the species tree as guide and iteratively merging

77 terminal sister taxa into ancestral genomes, the computational cost only grows

78 linearly with the number of genomes. However, reconstructing ancestral sequences

79 or profiles can prove challenging among distantly related species. Moreover, the

80 guide tree must be clearly defined and known, which is a problem for many

81 prokaryotes and eukaryotes.

82 Here, we present a new strategy for speeding up the all-against-all step, by first

83 clustering sequences and then performing an all-against-all within each cluster.

84 Crucially, in the clustering phase, the new approach accounts for residue-level

85 homology and not merely whole protein-level homology. As a result, sequences can

86 belong to multiple clusters. On sets of empirical bacterial and fungal protein

87 sequences, our new approach achieves a reduction in computational time of ~75%

88 while maintaining the sensitivity of all-against-all procedure (>99.6% of all pairs

89 identified by all-against-all).

2 Materials and Methods

91 We start by motivating our method under the assumption of perfect input data, then

92 present the modifications required to perform well on real, imperfect data. We then

93 describe the k-mer methods we used as points of comparison. Finally, we present

94 the empirical datasets used to evaluate the new method.

2.1   Building homologous clusters from perfect data

96 Typical downstream analysis of the all-against-all step requires an alignment score

97 for every significant pair of proteins (putative homologs). Therefore, aligning these

98 significant pairs seems difficult to avoid. In contrast, avoiding the alignment of

99 unrelated pairs can potentially reduce the overall runtime of the all-against-all. One

100 way to avoid some pairs is by using the transitive property of homology: typically—

101 we discuss pathological cases in the next section—if sequence A is homologous to

102 sequence B, and sequence B is homologous to sequence C, then sequences A and C

103 are homologous. A corollary of this is that if sequence A is homologous to sequence

104 B, and sequence A is *not* homologous to sequence C, then sequences B and C are

105 *not* homologous.

106 *Proof of corollary*: Let S1 denote "A is homologous to B", S2 "B is homologous

107 to C" and S3 "A is homologous to C". The transitivity of homology states that

108 $S1 \wedge S2 \Rightarrow S3$. By material implication, that it is equivalent to $\neg (S1 \wedge S2) \vee$

109 $S3$. De Morgan's law yields the equivalent expression $(\neg S1 \vee \neg S2) \vee S3$

110 which is, due to the associativity and commutativity of disjunction,

111 equivalent to $(\neg S1 \vee S3) \vee \neg S2$. Using the De Morgan's law again, the

112 expression transforms to the equivalent $\neg (S1 \wedge \neg S3) \vee \neg S2$. Finally, by

113 material implication it is equivalent to the expression $S1 \wedge \neg S3 \Rightarrow \neg S2$.

114 Therefore, in this particular scenario, if we infer that A and B are homologous and A

115 and C are non-homologous, there is no need to align sequences B and C. Another

116 way to consider this situation is that A acts as a representative for its homologous

117 sequence B. We can generalise this idea to more than three sequences by building

118 clusters of homologs for which a single sequence acts as a representative for all

119 members (trivial proof by induction).

120 Accordingly, we can split the all-against-all into two steps: (i) building homologous

121 clusters, and (ii) performing the all-against-all within each cluster. In the clustering

122 step, we initialise the procedure with a single cluster containing the first sequence

123 as its representative and sequentially consider all the other sequences: each

124 sequence is aligned to the representative sequence of each homologous cluster. If

125 an alignment score is significant, implying that the sequence is homologous to

126 members of that particular cluster, the sequence is added to the cluster. Else, a new

127 cluster is formed with that sequence as representative.

128 Under the assumptions that homology is transitive and that homology can be

129 perfectly ascertained through sequence alignment, this procedure guarantees that

130 all homologous sequences end up in a common cluster in $O(nk)$, where n is the

131 number of proteins and k is the number of clusters. Since $k \leq n$ (and possibly $k << n$),

132 this is more efficient than the $O(n^2)$ runtime of the standard all-against-all procedure.

133 In practice, however, homology at the level of proteins is not always transitive and

134 homology cannot be perfectly ascertained. Therefore, refinements to the basic idea

135 are required.

2.2 Building homologous clusters from real data: multiple representatives, multiple clusters, and subsequence-level homology

139 The two assumptions underlying the algorithm for ideal data are not tenable on real

140 data. First, highly distant (and thus dissimilar) homologous sequences can be very

141 difficult or impossible to reliably identify. Thus, if the representative is very distant to

142 a particular homologous sequence, homology can nevertheless be missed. Second,

143 homology does not necessarily apply uniformly across all residues of a protein. For

144 transitivity to hold, it is essential that homology applies to a common set of

145 residues. However, insertions and gene fusion can lead to situations where this is

146 not the case (see Fig. 1A for an illustrative example).

147 Thus, to deal more effectively with real sequences, we modify the basic algorithm in

148 three ways. First, instead of a single representative sequence, we also consider a

149 variation of the algorithm that uses several representative sequences for each

150 cluster. This can be advantageous for homologous clusters containing very dissimilar

151 sequences, which are thus poorly represented by any single sequence. Second, we

152 allow for the possibility that some sequences might be homologous to more than

153 one cluster. This could be the case if homologous clusters are fragmented due to

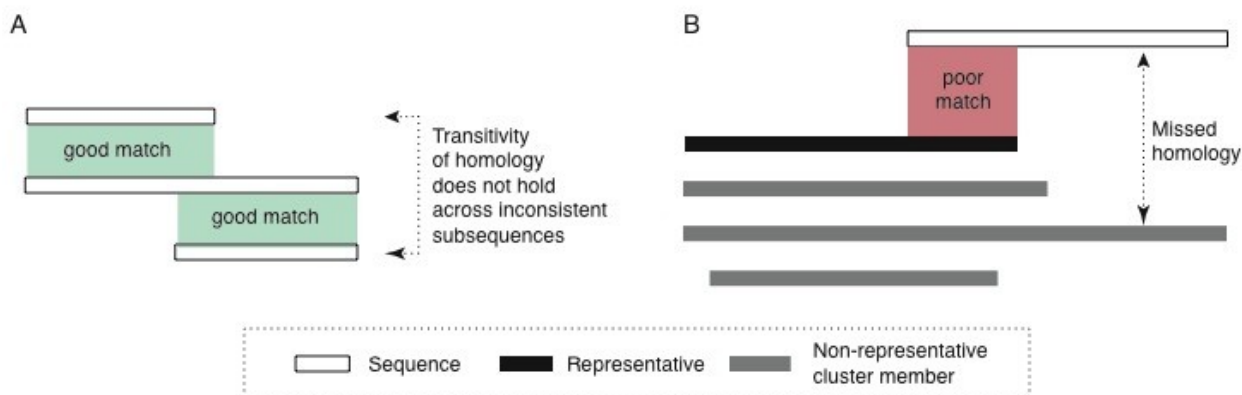
154 high divergence among some of their members (and among their representatives in

155 particular). This also enables multi-domain proteins to be included in clusters

156 corresponding to different domains. Third, we take into account subsequence-level

157 homology: we ensure that the entire length of a particular protein (minus a tolerance

158 parameter, set to 20 amino-acid residues in this study) is covered by the cluster(s)

159 to which the sequence belongs. If this is not the case, a new cluster is created with
160 the sequence as representative. Note that this is in addition to inclusion into any
161 cluster with significant alignment (Fig. 1B)).

162 Figure 1: **Diagram of potential problems with transitivity of homology**. (A) The
163 transitivity of homology does not hold if the residues involved are inconsistent. (B)
164 Homologous relationships can be missed if significant sequence parts are not covered by the
165 representative sequence. To address this problem, our approach would create a new cluster,
166 with the sequence that is not fully covered as its representative.

167 Putting these ideas together, our approach for building homologous clusters is as
168 follows. First, we sort the input genomes by their number of sequences in
169 descending order. For every genome, starting with the one with the highest number
170 of sequences, we process its sequences one by one as they are listed in the
171 database file. As there are no clusters at the beginning of the process, the first
172 sequence founds a cluster and becomes its representative. Every subsequent
173 sequence is aligned with the representatives of the current cluster(s) using Smith-
174 Waterman (1981). We run the vectorised Smith-Waterman implementation of
175 Szalkowski *et al.* (2008) using the 224 GCB scoring matrix (Gonnet et al. 1992). If
176 the score is above the minimum threshold of 135.75, the sequence is added to the
177 cluster. Furthermore, if the number of cluster representatives in the particular
178 cluster is below the maximum allowed, the newly added sequence becomes a
179 representative. We also keep track how much the sequence is covered by the
180 representatives of the assigned clusters. We do not introduce any restrictions on the
181 size of the clusters nor on the number of clusters that a sequence is assigned to.
182 After exhaustive search through all cluster representatives, we assess whether the
183 sequence was added to one or multiple clusters. If the full length of the sequence

184 (minus a tolerance of 20 amino acids) is not covered by the clusters to which the

185 sequence was added, an additional cluster is created with the sequence as a

186 representative. The same applies if the sequence was not assigned to any clusters

187 (see Box 1 for pseudocode).

188

```
Input: Genomes , thresholds T=135.75 (minimum score), L (number of
representatives per cluster) and C=20 (minimum coverage)

Clusters ← ∅
for each genome in Genomes do
        for each sequence in genome do
              if Clusters = ∅ then
                        initialise newCluster
                        newCluster ← sequence
                        newCluster.representatives ← sequence
                        Clusters.append(newCluster)
              else
                        for each cluster in Clusters do
                                for each representative in cluster.representatives do
                                        if alignment-score(sequence, representative) > T
then
                                                cluster.append(sequence)
                                                if |cluster.representatives| < L then
                                                        cluster.representatives ← sequence
                                                end if
                                        end if
                                end for
                        end for
              end if
              if sequence was assigned to at least one cluster then
                        coverages ← ∅
                        for each representative of the assigned clusters do
                                coverages.append(coverage(sequence, representative))
                        end for
              end if
              if sequence was not assigned to any cluster or max(coverages) < C
      then
                        initialise newCluster
                        newCluster ← sequence
                        newCluster.representatives ← sequence
                        Clusters.append(newCluster)
              end if
        end for
end for

Output: a set of homologous clusters Clusters
```

189   Box 1. **Pseudocode of the proposed cluster building procedure.**

2.3 Computing all-against-all within each cluster

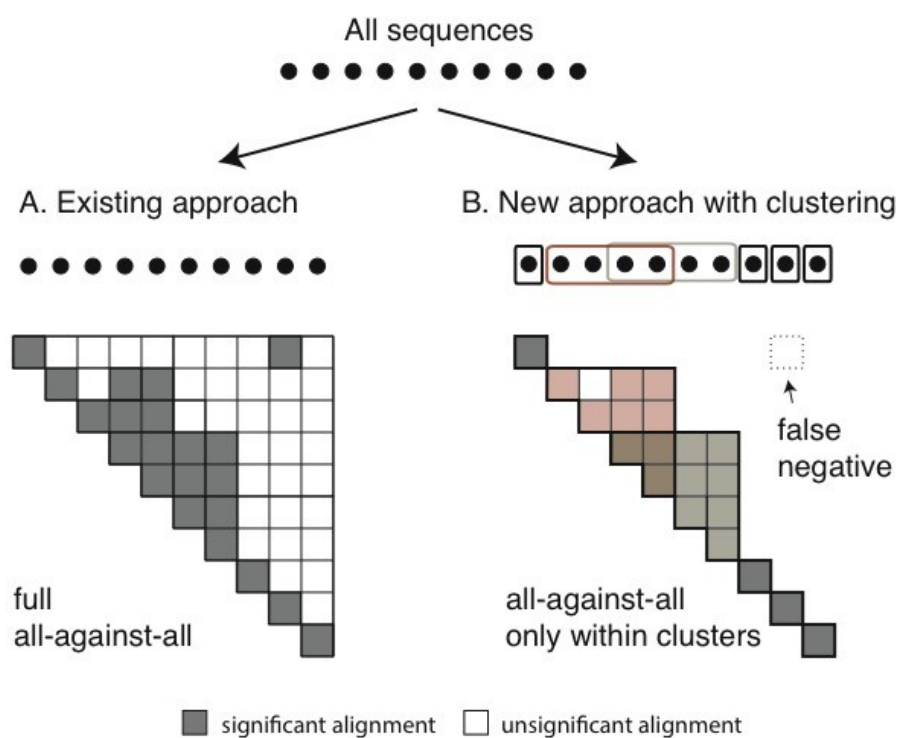
190

191 Once we have identified all homologous clusters, we compute an all-against-all

192 within each cluster using the same criteria as the global all-against-all (Figure 2).

193 This ensures that all retained pairs agree with the score and length requirements of

194 the full all-against-all, and provides scores for all homologous pairs, which are often

195 required for downstream analyses.

196 Figure 2. **Comparison between the current and the new approach.** Diagram of (A) the
197 current OMA all-against-all and (B) the new approach with homologous clustering and
198 sequence coverage. In the proposed approach, the sequences are first clustered and then
199 the all-against-all is run only within clusters. The overall number of computations is
200 considerably reduced, but some homologous relationships can be missed.

2.4 Comparison with full all-against-all and other methods

201

202 The baseline of this study is given by the current all-against-all procedure in the

203 OMA database (Roth et al. 2008; Altenhoff et al. 2011). It runs a highly optimised

204 Smith-Waterman (1981) local alignment implementation (Szalkowski et al. 2008)

205 using the 224 GCB scoring matrix (Gonnet et al. 1992). To be inferred as homologs,

206 sequence pairs need to reach an alignment score of at least 135.75 with the 224

207 GCB matrix (and 181 with an optimised GCB matrix, but this additional requirement

208 was not considered in this study), with the additional requirement that the length of

209 the shorter sequence aligned be at least 61% of the longer sequence. These

210 parameters were previously established to yield good results on empirical data (Roth

211 et al. 2008).

212 To compare the performance of the new approach to *k*-mer clustering, we also ran

213 kClust (Hauser et al. 2013) and UCLUST (Edgar 2010). We ran kClust with default set

214 of parameters, where the required minimum sequence identity was 30% (`-s 1.12`)

215 and the minimum alignment coverage of the longer sequence was 80% (`-c 0.8`). To

216 increase sensitivity, we also ran kClust with 30% as minimum for sequence identity

217 (`-s 1.12`) and 50% as minimum alignment coverage (`-c 0.5`). The jobs were run on

218 a single core of a Dell R910 (Intel Xeon E7-8837, 2.66 GHz) with 32 cores and 1TB of

219 memory. For UCLUST, we used the recommended option `-cluster_fast`. We first

220 ran UCLUST with default parameters, specifying 30% as sequence identity (`-id`

221 `0.3`). We also ran it with the additional requirements on coverage of the target

222 sequence being 50% (`-target_cov 0.5`) and allowing sequences to be assigned to

223 multiple clusters (`-maxaccepts 0`), without limiting the number of sequences

224 rejected before quitting search (`-maxrejects 0`). In both cases, only sequences

225 longer than 50 amino acid residues were taken into account—the same length

226 requirement as in the all-against-all procedure. UCLUST was run on a MacBook Air

227 (Intel Core i7, 1.7 GHz) dual-core with 8GB of memory.

228 **2.5 Datasets**

229 We analysed the performance of the new approach on two distinct datasets:

230 bacteria and fungi. Bacteria dataset contained 14 proteomes, fungi contained 12, all

231 sampled from the OMA database (Altenhoff et al. 2011) (March 2014 release; see

232 *Supplementary File* 1 for complete list). The distribution of sequence lengths and

233 evolutionary distances among significant pairs is provided in *Supplementary*

234 *materials* Fig. 1-2. To assess the scaling behaviour of each variant, we ran them on

235 subsets which were formed as follows: all genomes in the dataset were sorted by

236 their size and the ones in the middle of the list were chosen to form a subset. By

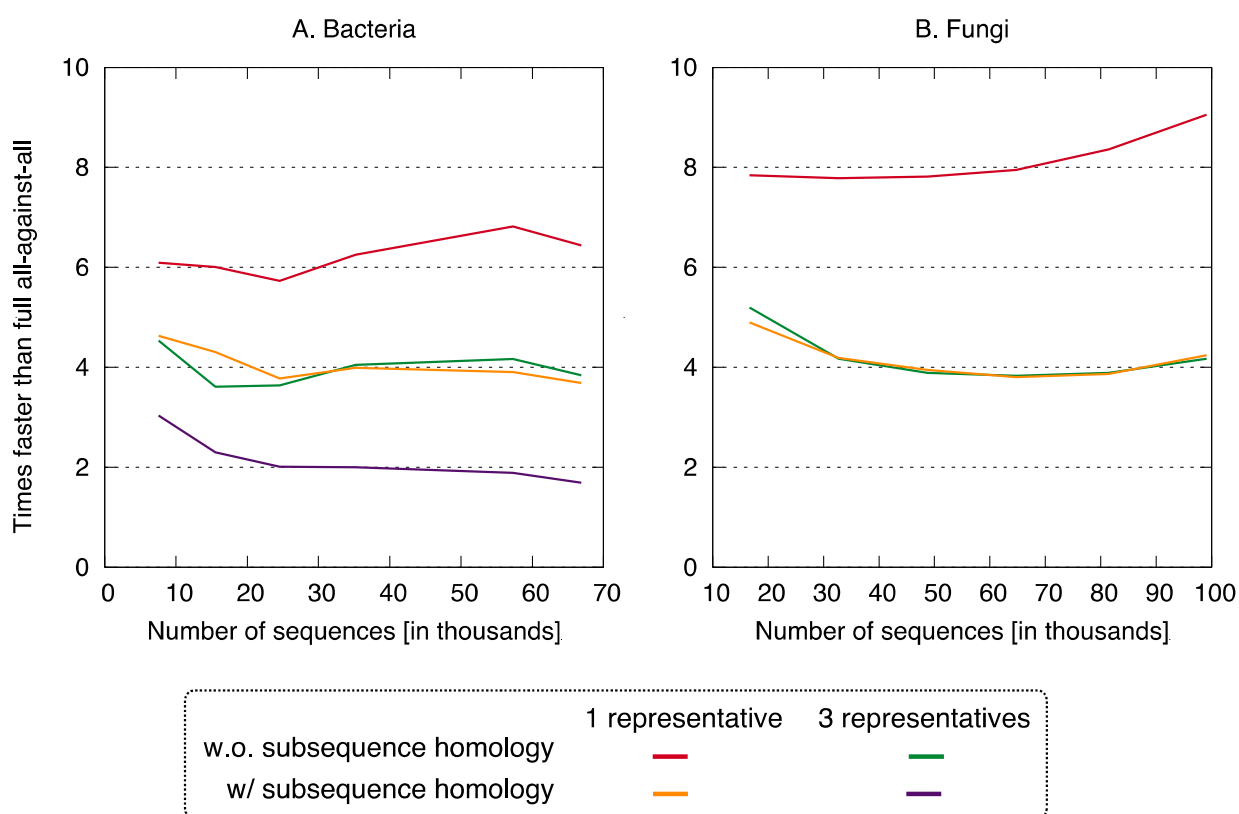
237 choosing the central 2, 4, 6, 8, 12 and finally all 14 proteomes, subsets of bacteria

238 were constructed. We repeated the procedure for fungi and obtained subsets of size

239 2, 4, 6, 8, 10 and 12 proteomes.

240   In addition, we performed an extra analysis on a diverse dataset containing two
241   vertebrates, one plant, one protist, one fungus, and one bacterium (*Supplementary*
242   *File* 1).

3 Results

244   To assess the time and accuracy of our new approach, we compared the four
245   clustering variants—one or three cluster representatives, each combined with either
246   taking into account residue-level homology or not (see Methods)—on empirical
247   bacterial genomes. We also repeated the three fastest variants on empirical fungal
248   datasets.

249   Figure 3: **Speed up using the new approach.** The proposed approach is 2-9x faster,
250   depending on datasets and variants. On the fungi dataset, only the three fastest variants
251   were computed.

252   All four clustering variants decreased runtime compared to the full all-against-all
253   algorithm, with a speedup factor of 2-9 depending on the variant and dataset (Fig 3).

254 Unsurprisingly, the reduction was strongest in variants considering a single

255 representative sequence per cluster. Likewise, ignoring residue-level homology lead

256 to faster runtime, though the difference was more modest.

257 In terms of performance—which we measured in terms of the proportion of

258 significant pairs from the full all-against-all that were recovered by the new variants

259 (referred to as "recall") or its complement (referred to as "missing pairs")—we

260 observed a clear benefit in taking into account residue-level homology, with recall

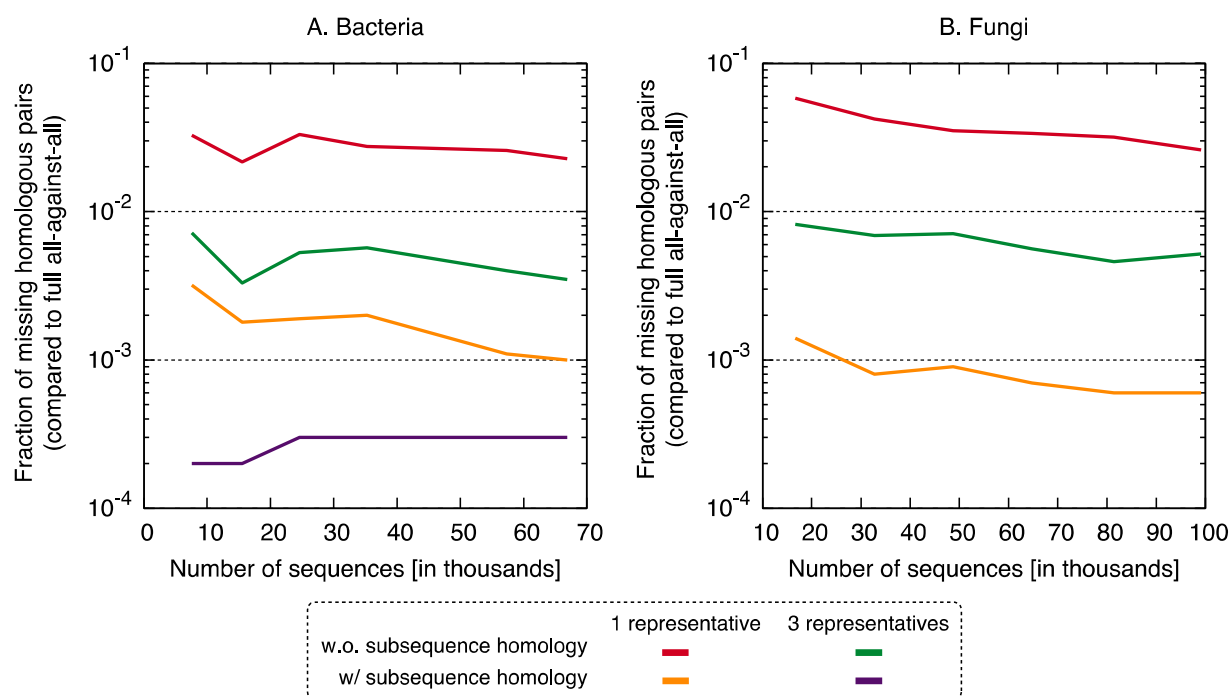
261 values consistently >99.6% (Fig. 4). The use of three representative sequences per

262 cluster also led to increase in recall, but not nearly to the same extent. The recall

263 values obtained for all variants were consistent across the bacteria and fungi

264 datasets and for different sizes, which suggests that they might hold across other

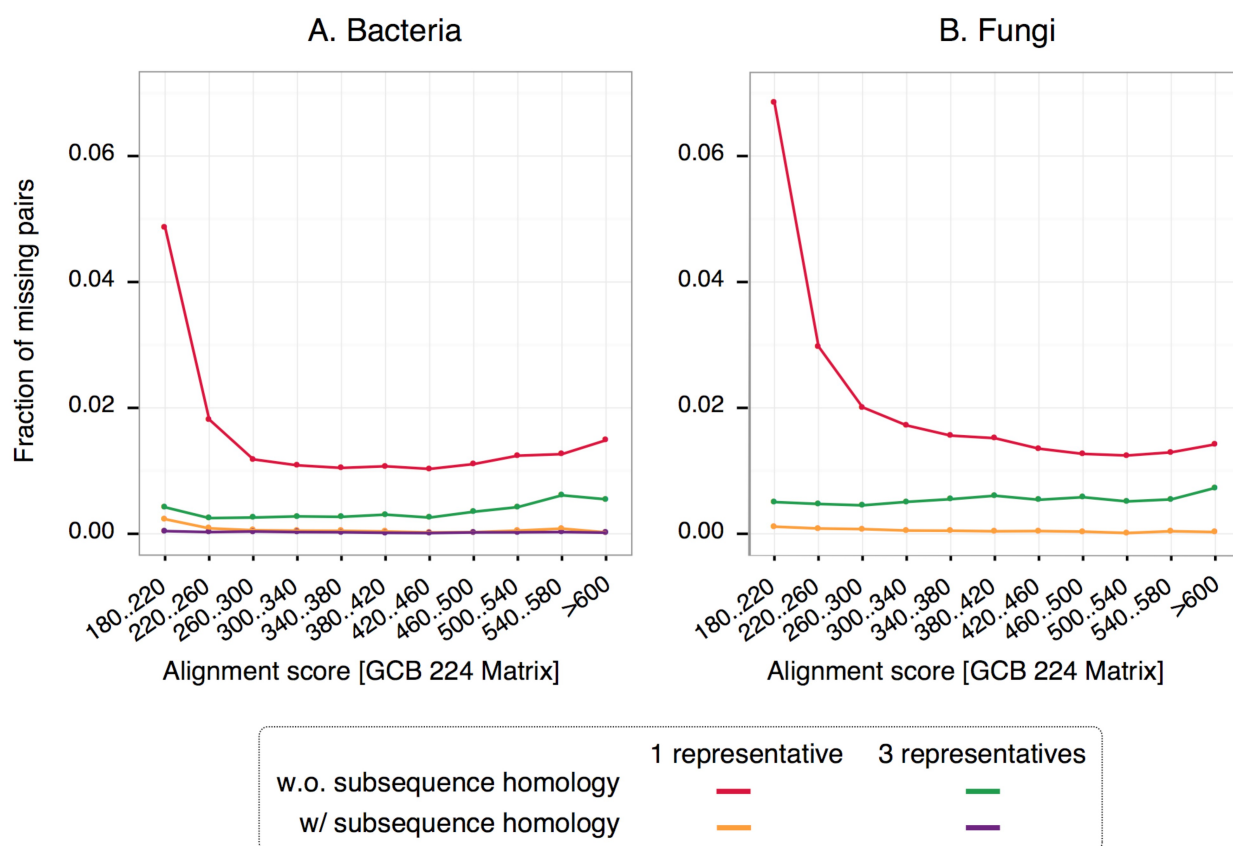
265 datasets as well. To illustrate the nature of missing homologs, two cases are detailed

266 in the *Supplementary materials* (*Supplementary materials* Fig. 3-6).

267 Figure 4: **Fraction of the pairs missed when using the proposed algorithm.** The new
268 approach misses 0.4-6% of the pairs from the full all-against-all in its simple variant (1 and 3
269 representatives), and 0.01-0.3% taking into account sub-sequence homology.

270 Putting runtime and accuracy together, the best results were obtained by taking into

271 account subsequence-level homology using a single representative sequence, which

272 achieved a ~4x speedup while maintaining >99.6% recall in both datasets.

273 To gain insight into the nature of missing pairs, we analysed the distribution of
274 alignment scores of missing pairs compared to that of all significant matches (Fig.
275 5). For all method variants, but especially those taking into account subsequence-
276 level homology, the distribution is heavily skewed toward low-scoring matches. This
277 is favourable, because low-scoring matches are less reliable in the first place and
278 therefore downstream analyses are already designed to cope with missing
279 homologous relationships among distant pairs (e.g. Altenhoff et al. 2013). We also
280 note that the higher recall observed with three representatives instead of one is
281 mainly concentrated on low scoring matches. Furthermore, a per-gene analysis of
282 the rate of missing pairs suggests that errors are distributed quite evenly across all
283 family sizes (*Supplementary materials* Fig. 7).

284 Figure 5: **Distribution of alignments scores of missing pairs compared with all pairs**
285 **("background").** With a single representative, the distribution is heavily skewed toward
286 low-scoring pairs.

287 For any given taxonomic range, adding new genomes and the corresponding new
288 sequences can only increase the number of clusters. However, as the number of

PeerJ Reviewing Manuscript

289 genomes grows, we can expect that an increasing proportion of the sequences will
290 fall into one of the existing clusters. Thus, we should see a tapering in the number of
291 clusters as a function of the number of sequences, which would be favourable in
292 terms of runtime of the algorithm. In our datasets, we could not observe such
293 tapering, and instead the growth in cluster numbers was broadly linear
294 (*Supplementary materials* Fig. 8-9). This suggests that 12-14 genomes are too few to
295 discern the additional asymptotic benefits of our new approach. A similar conclusion
296 can be drawn from the distribution of cluster size, which is heavily skewed toward
297 very small clusters (*Supplementary materials* Fig. 10). The large overlap among
298 numerous clusters and the existence of many sequences included in multiple
299 clusters (*Supplementary materials* Fig. 11) suggests improvement potential by
300 merging some of the clusters (see also *Discussion* below).

301 To assess the impact of the new clustering approach on orthology inference, we ran
302 the OMA standalone software (Altenhoff et al. 2013) using our best new variant (1
303 representative, with subsequence homology) compared to the full all-against-all as a
304 reference. On the largest Bacteria and Fungi dataset, the proportion of orthologs
305 that were recovered was 99.71% and 99.87% respectively. This is slightly lower than
306 the proportion of recovered homologs (99.9% and 99.94%), but remains very high.

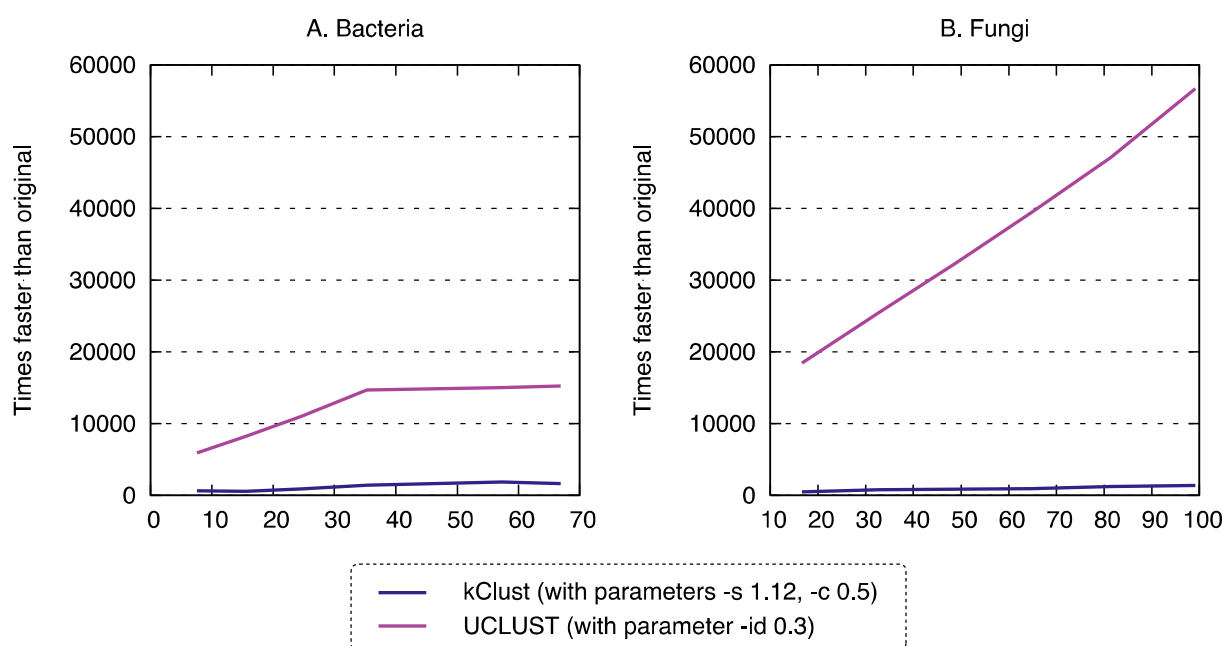
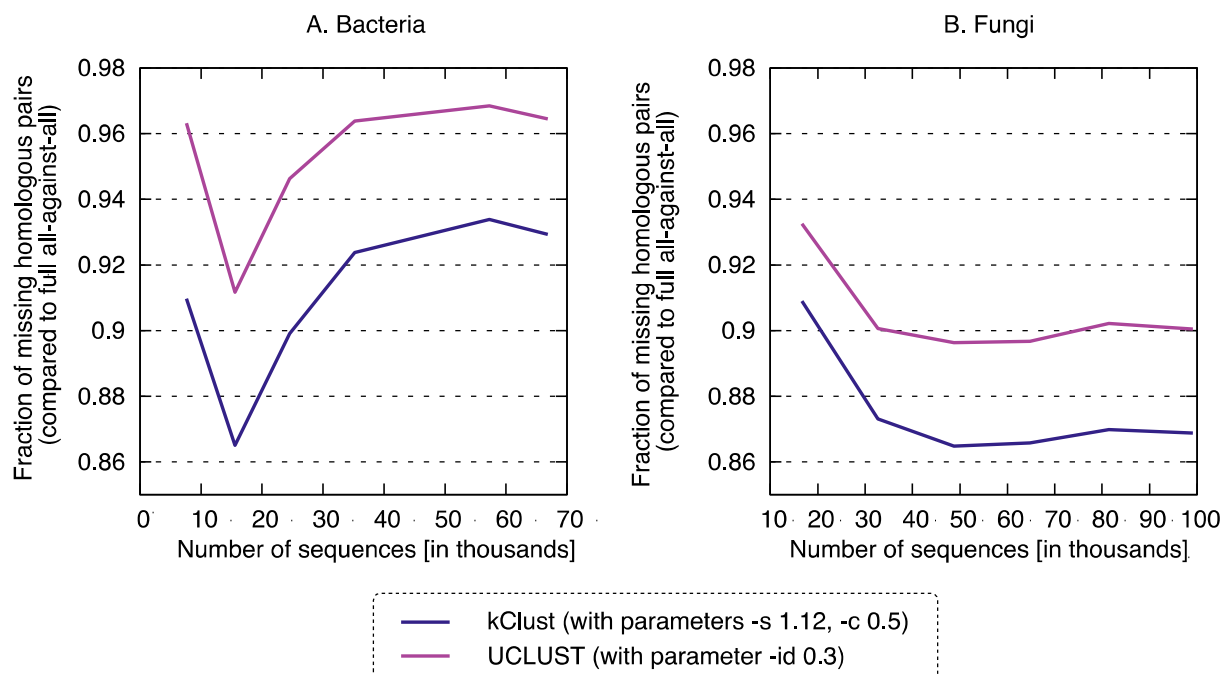
307 Furthermore, to test whether large genomes and/or multidomain proteins could
308 affect the performance of the new approach, we conducted an additional
309 comparison on a genome set containing two vertebrates (Human and *Xenopus*
310 *tropicalis*), one plant (*Arabidopsis thaliana*) and three unicellular organisms
311 (*Plasmodium falciparum, Saccharomyces cerevisiae*, and *Bacillus subtilis*). We
312 applied the variant with one representative and subsequence homology. It resulted
313 in a considerably higher speedup factor of 12.05, with a recall of 99.94%. This
314 suggests that the new approach behaves well in the presence of large genomes and
315 numerous multi-domain proteins as well.

316 Finally, we sought to compare the new approach to the state-of-the-art *k*-mer
317 clustering approaches UCLUST [(Edgar 2010)](#) and kClust [(Hauser et al. 2013)](#). As
318 elaborated in the introduction, these methods have been developed to cluster
319 closely related sequences, but given their similar conceptual ideas (clustering, use
320 of representative sequences), they provide an interesting point of comparison. As

321 expected, these approaches were orders of magnitude faster (speedup 500-1900x

322 for kClust, over 6000x for UCLUST; Fig. 6), but their recall was typically in the single

323 percentage points (6.62-13.52% for kClust, 3.16-10.37% for UCLUST), even with

324 inclusive parameters (Fig. 7; *Supplementary File* 2).

325 Figure 6: **Runtime of typical *K*-mer approaches.** kClust and UCLUST are several orders of
326 magnitude faster than full all-against-all.

327   Fig. 7. **Recall of typical K-mer approaches.** kClust and UCLUST only recover 3-14% of the
328   homologous pairs identified by the full all-against-all procedure, even when they are run with
329   sensitive parameter settings (see *Materials and Methods* for details).

4 Discussion and outlook

331   The all-against-all phase is at the basis of many orthology inference algorithms but

332   due to its quadratic time complexity in terms of number of sequences, the all-

333   against-all phase can be a bottleneck. This work suggests that the transitivity of

334   homology can be exploited to substantially speedup homology inference while

335   maintaining sensitivity—provided homology is defined in terms of consistent

336   subsequence stretches.

337   The datasets used in this study are challenging, with median distance of 146 PAM

338   (1.46 substitution per site on average) in the bacteria dataset and 149 PAM in the

339   fungi dataset (*Supplementary materials* Fig. 2). This implies that half of homologous

340   pairs have less than 30% sequence identity. At such high levels of divergence, *k*-mer

341   based methods perform poorly compared to all-against-all dynamic programming

342   alignment. In contrast, our transitivity clustering approach performed well. It is likely

343   to perform even better on evolutionarily closer sets of taxa, such as vertebrate

344   species or flowering plant species.

345   The speedup we observed with the new approach is substantial. But because the

346   number of clusters increased roughly linearly with the number of genomes in our

347   datasets of up to 14 genomes, the overall time complexity still grows quadratically

348   in this range. However, as rarefaction curves show (e.g. Mira et al. 2010), the growth

349   in the number of homologous clusters typically tapers off. Thus, it is possible that

350   the asymptotic complexity of the new procedure is subquadratic in the number of

351   species. This will need to be confirmed in future work.

352   Apart from the full dynamic programming alignment comparison with cluster

353   representatives and the subsequence-level homology consideration, our approach

354   distinguishes itself from most clustering methods in that it allows sequences to

355   match to several clusters. If homology was always uniform across the whole

356   sequence length and could be perfectly inferred from present-day sequences, each

357   sequence would belong to one and only one homologous cluster. In practice

358 however, these assumptions do not hold: gene fusion can result in sequences that
359 have homologous relationships across multiple clusters; some homologs have
360 diverged too far to be inferred as such, thus resulting in cluster fragmentation. By
361 allowing for sequences to belong to multiple clusters, our method is robust to these
362 complications.

363 Though the present study focuses on Smith-Waterman dynamic programming
364 alignments, a similar clustering approach would be possible with the faster but less
365 sensitive BLAST (Altschul et al. 1997). One complication with BLAST is however that
366 the sequence database needs to be re-indexed whenever a sequence is added. To
367 mitigate this, one could add new representative sequences in batches, with the
368 additional complication that sequences within each batch would also need to be
369 aligned to one another.

370 A further advantage of the clustering approach is that it also works well in the
371 context of "semi-curated" databases, such as in COGs (Tatusov et al. 2003), Panther
372 (Mi et al. 2013), or PFam (Finn et al. 2014). Indeed, it is conceptually straightforward
373 to let curators optimise particular homologous clusters by fine-tuning representative
374 sequences, coverage and score thresholds on a cluster-by-cluster basis.

375 These promising results notwithstanding, the new approach still has much potential
376 for improvement. To further improve the recall, several ideas could be explored.
377 First, the procedure could merge clusters that have a high proportion of members in
378 common. This could recover some of the missed homologous pairs. Second, the
379 choice of cluster representatives could be improved. Indeed, the current strategy of
380 selecting the first sequence (or the first three sequences) added to the cluster is
381 likely to be suboptimal in most instances. Instead, a better choice of representative
382 would be to try to select a sequence with minimal average distance to all cluster
383 members (a "centroid" sequence). Another idea, suggested by Dr. Ikuo Uchiyama in
384 his peer-review, would be to vary the number of representatives depending on the
385 particular needs of each cluster, e.g. by adding a new member as a representative if
386 its similarity score with the existing representative(s) is below a certain threshold.
387 Third, as alternative to representative sequences, clusters could be represented by
388 Hidden Markov model profiles. Such approach, at the basis of well-established
389 methods such as PSI-Blast (Altschul et al., 1997), HMMER (Eddy, 2011), or HHBlitz

390 (Remmert et al., 2011), is likely be more sensitive than pairwise alignment with a
391 single representative sequence. The profiles could be periodically updated as new
392 sequences are assigned to the cluster.

393 We also see potential to further improve the speed of the new approach. First,
394 merging homologous clusters can lead to a speed improvement because the cost of
395 assigning sequences to clusters grows linearly in the number of clusters. This needs
396 to be done carefully, because excessive merging—the merging of clusters containing
397 a substantial number of non-homologous pairs—can reduce the efficiency of the
398 within-cluster all-against-all, whose time complexity grows quadratically in the
399 number of sequences. Second, it may be possible to optimise the assignment of
400 sequences to clusters by identifying clusters that are so different to one another that
401 they are practically mutually exclusive and thus inclusion into one implies exclusion
402 from the other. An empirical way of establishing such mutual exclusivity would be to
403 keep track of the number or proportion of sequences belonging to both clusters.
404 Finally, the new approach could be parallelised. One way of parallelising the
405 assignment of sequences to clusters would be to use a Publisher-Subscriber model
406 (Eugster et al. 2003): a "master" process would start the analysis of a new sequence
407 by distributing it to a set of "workers", each responsible to compare the sequence to
408 a subset of all existing clusters. Each worker would thus align the new sequence to
409 its designated subset of clusters and reports back significant matches and their
410 associate sequence ranges (subsequence-level coverage). Once the master process
411 has received this information from all workers, it would ensure that the new
412 sequence is fully covered by matches to existing clusters, and else it would generate
413 a new cluster with that sequence as representative. As for the within-cluster all-
414 against-all comparisons, they could be straightforwardly parallelised thanks to the
415 lack of dependency among all pairs.

416 Meanwhile, a serial implementation of our best variant (accounting for subsequence-
417 level homology using a single representative sequence) is available as part of the
418 open source OMA standalone package (http://omabrowser.org/standalone).

Acknowledgements

419

420 This article is dedicated to Gaston H. Gonnet on the occasion of his retirement. We

421 thank Kevin Gori, Henning Redestig, Nives Škunca, Bartlomiej Tomiczek for useful

422 feedback on the manuscript, as well as Odile Lecompte, Ikuo Uchiyama and one

423 anonymous reviewers for useful feedback on the manuscript.

Supplementary Materials

425 ● PDF containing supplementary figures 1-11 (appended to this preprint for

426 convenience)

427 ● Supplementary File 1: list of species, including taxonomic ID and genome source and

428 version.

429 ● Supplementary File 2: detailed comparisons with kClust and UCLUST.

Bibliography

431 Altenhoff A.M., Dessimoz C. 2012. Inferring Orthology and Paralogy. In: Anisimova M.,
432   editor. Evolutionary Genomics. Humana Press. p. 259–279.
433 Altenhoff A.M., Gil M., Gonnet G.H., Dessimoz C. 2013. Inferring hierarchical
434   orthologous groups from orthologous gene pairs. PLoS One. 8:e53786.
435 Altenhoff A.M., Schneider A., Gonnet G.H., Dessimoz C. 2011. OMA 2011: orthology
436   inference among 1000 complete genomes. Nucleic Acids Res. 39:D289–94.
437 Altenhoff A.M., Studer R. a., Robinson-Rechavi M., Dessimoz C. 2012. Resolving the
438   ortholog conjecture: orthologs tend to be weakly, but significantly, more similar
439   in function than paralogs. PLoS Comput. Biol. 8:e1002514.
440 Altschul S.F., Madden T.L., Schäffer A.A., Zhang J., Zhang Z., Miller W., Lipman D.J.
441   1997. Gapped BLAST and PSI-BLAST: a new generation of protein database
442   search programs. Nucleic Acids Res. 25:3389–3402.
443 DeLuca T.F., Cui J., Jung J.-Y, St Gabriel K.C., Wall D.P. 2012. Roundup 2.0: enabling
444   comparative genomics for over 1800 genomes. Bioinformatics. 28:715–716.
445 Dessimoz C., Cannarozzi G., Gil M., Margadant D., Roth A., Schneider A., Gonnet G.
446   2005. OMA, A Comprehensive, Automated Project for the Identification of
447   Orthologs from Complete Genome Data: Introduction and First Achievements.
448   RECOMB 2005 Workshop on Comparative Genomics.61–72.
449 Dessimoz C., Gabaldón T., Roos D.S., Sonnhammer E.L.L., Herrero J., Quest for
450   Orthologs Consortium. 2012. Toward community standards in the quest for
451   orthologs. Bioinformatics. 28:900–904.
452 Edgar R.C. 2010. Search and clustering orders of magnitude faster than BLAST.
453   Bioinformatics. 26:2460–2461.
454 Eugster P.T., Felber P.A., Guerraoui R., Kermarrec A.-M. 2003. The Many Faces of
455   Publish/Subscribe. ACM Comput. Surv. 35:114–131.
456 Finn R.D., Bateman A., Clements J., Coggill P., Eberhardt R.Y., Eddy S.R., Heger A.,
457   Hetherington K., Holm L., Mistry J., Sonnhammer E.L.L., Tate J., Punta M. 2014.
458   Pfam: the protein families database. Nucleic Acids Res. 42:D222–30.
459 Fitch W.M. 1970. Distinguishing homologous from analogous proteins. Syst. Zool.
460   19:99–113.

461   Gabaldón T., Koonin E.V. 2013. Functional and evolutionary implications of gene
462   orthology. Nat. Rev. Genet. 14:360–366.
463   Gaudet P., Livstone M.S., Lewis S.E., Thomas P.D. 2011. Phylogenetic-based
464   propagation of functional annotations within the Gene Ontology consortium.
465   Access. 12.
466   Gonnet G.H., a Cohen M., a Benner S. 1992. Exhaustive matching of the entire
467   protein sequence database. Science. 256:1443–1445.
468   Hauser M., Mayer C.E., Söding J. 2013. kClust: fast and sensitive clustering of large
469   protein sequence databases. BMC Bioinformatics. 14:248.
470   Li W., Godzik A. 2006. Cd-hit: a fast program for clustering and comparing large sets
471   of protein or nucleotide sequences. Bioinformatics. 22:1658–1659.
472   Lynch M. 2000. The Evolutionary Fate and Consequences of Duplicate Genes.
473   Science. 290:1151–1155.
474   Metzker M.L. 2009. Sequencing technologies — the next generation. Nat. Rev.
475   Genet. 11:31–46.
476   Mi H., Muruganujan A., Thomas P.D. 2013. PANTHER in 2013: modeling the evolution
477   of gene function, and other gene attributes, in the context of phylogenetic
478   trees. Nucleic Acids Res. 41:D377–86.
479   Mira A., Martín-Cuadrado A.B., D'Auria G., Rodríguez-Valera F. 2010. The bacterial
480   pan-genome: a new paradigm in microbiology. International Microbiology.
481   13:45–57.
482   Pagani I., Liolios K., Jansson J., Chen I. 2012. The Genomes OnLine Database (GOLD)
483   v. 4: status of genomic and metagenomic projects and their associated
484   metadata. Nucleic acids.
485   Patterson C. 1988. Homology in classical and molecular biology. Mol. Biol. Evol.
486   5:603–625.
487   Powell S., Forslund K., Szklarczyk D., Trachana K., Roth A., Huerta-Cepas J., Gabaldón
488   T., Rattei T., Creevey C., Kuhn M., Jensen L.J., von Mering C., Bork P. 2013.
489   eggNOG v4.0: nested orthology inference across 3686 organisms. Nucleic Acids
490   Res. 42:D231–9.
491   Rentzsch R., Orengo C. a. 2009. Protein function prediction--the power of multiplicity.
492   Trends Biotechnol. 27:210–219.
493   Roth A.C.J., Gonnet G.H., Dessimoz C. 2008. Algorithm of OMA for large-scale
494   orthology inference. BMC Bioinformatics. 9:518.
495   Schreiber F., Sonnhammer E.L.L. 2013. Hieranoid: hierarchical orthology inference. J.
496   Mol. Biol. 425:2072–2081.
497   Smith T.F., Waterman M.S. 1981. Identification of common molecular subsequences.
498   J. Mol. Biol. 147:195–197.
499   Sonnhammer E.L.L., Gabaldón T., Sousa da Silva A.W., Martin M., Robinson-Rechavi
500   M., Boeckmann B., Thomas P.D., Dessimoz C., the Quest for Orthologs
501   consortium. 2014. Big data and other challenges in the quest for orthologs.
502   Bioinformatics. Advanced access online.
503   Szalkowski A., Ledergerber C., Krähenbühl P., Dessimoz C. 2008. SWPS3 - fast multi-
504   threaded vectorized Smith-Waterman for IBM Cell/B.E. and x86/SSE2. BMC Res.
505   Notes. 1:107.
506   Tatusov R.L., Fedorova N.D., Jackson J.D., Jacobs A.R., Kiryutin B., Koonin E.V., Krylov
507   D.M., Mazumder R., Mekhedov S.L., Nikolskaya A.N., Sridhar Rao B., Smirnov S.,
508   Sverdlov A.V., Vasudevan S., Wolf Y.I., Yin J.J., Natale D. a. 2003. The COG
509   database: an updated version includes eukaryotes. BMC Bioinformatics. 4:41.
510   Tatusov R.L. 1997. A Genomic Perspective on Protein Families. Science. 278:631–
511   637.
512   Vilella A.J., Severin J., Ureta-Vidal A., Heng L., Durbin R., Birney E. 2008.
513   EnsemblCompara GeneTrees: Complete, duplication-aware phylogenetic trees

514   in vertebrates. Genome Res. 19:327–335.

515   Waterhouse R.M., Tegenfeldt F., Li J., Zdobnov E.M., Kriventseva E.V. 2013. OrthoDB:

516   a hierarchical catalog of animal, fungal and bacterial orthologs. Nucleic Acids

517   Res. 41:D358–65.

Figure 1

Fig 1: Diagram of potential problems with transitivity of homology.

(A) The transitivity of homology does not hold if the residues involved are inconsistent. (B) Homologous relationships can be missed if significant sequence parts are not covered by the representative sequence. To address this problem, our approach would create a new cluster, with the sequence that is not fully covered as its representative.

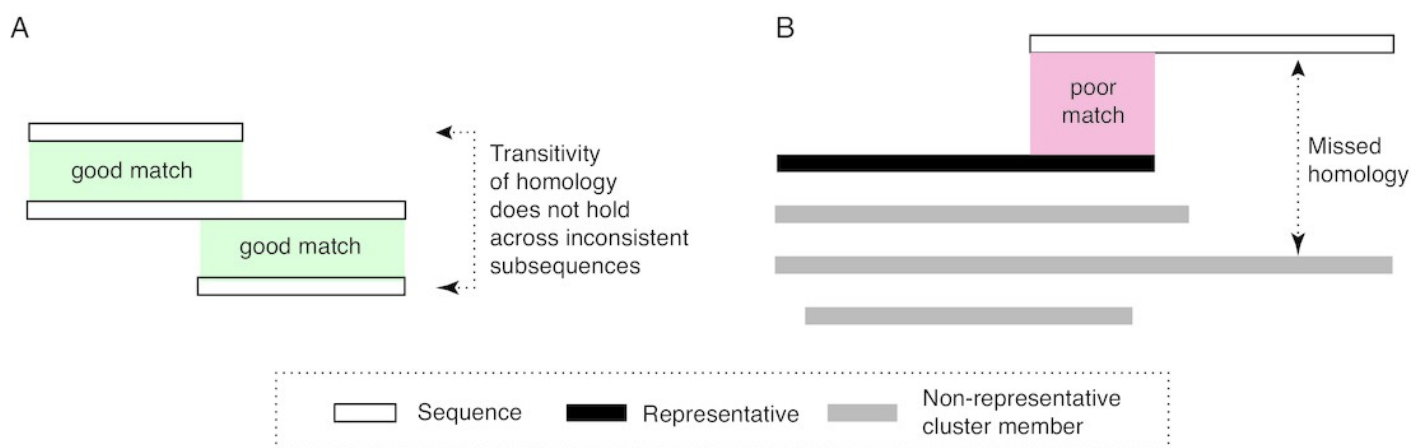



Figure 2

Fig. 2: Comparison between the current and the new approach.

Diagram of (a) the current OMA all-against-all step and (b) the new approach with homologous clustering and sequence coverage. In the proposed approach, the sequences are first clustered and then the all-against-all is run only within clusters. The overall number of computations is considerably reduced, but some homologous relationships can be missed.

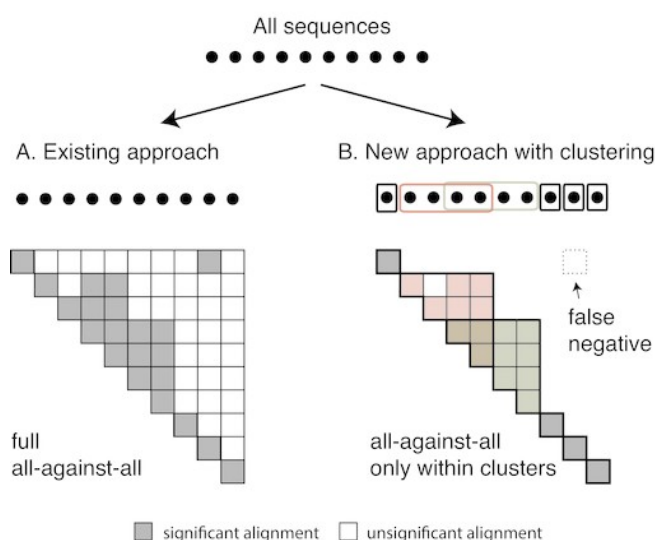



Figure 3

Fig. 3: Speed up using the new approach.

The proposed approach is 2-9x faster, depending on datasets and variants. On the fungi dataset, only the three fastest variants were computed.

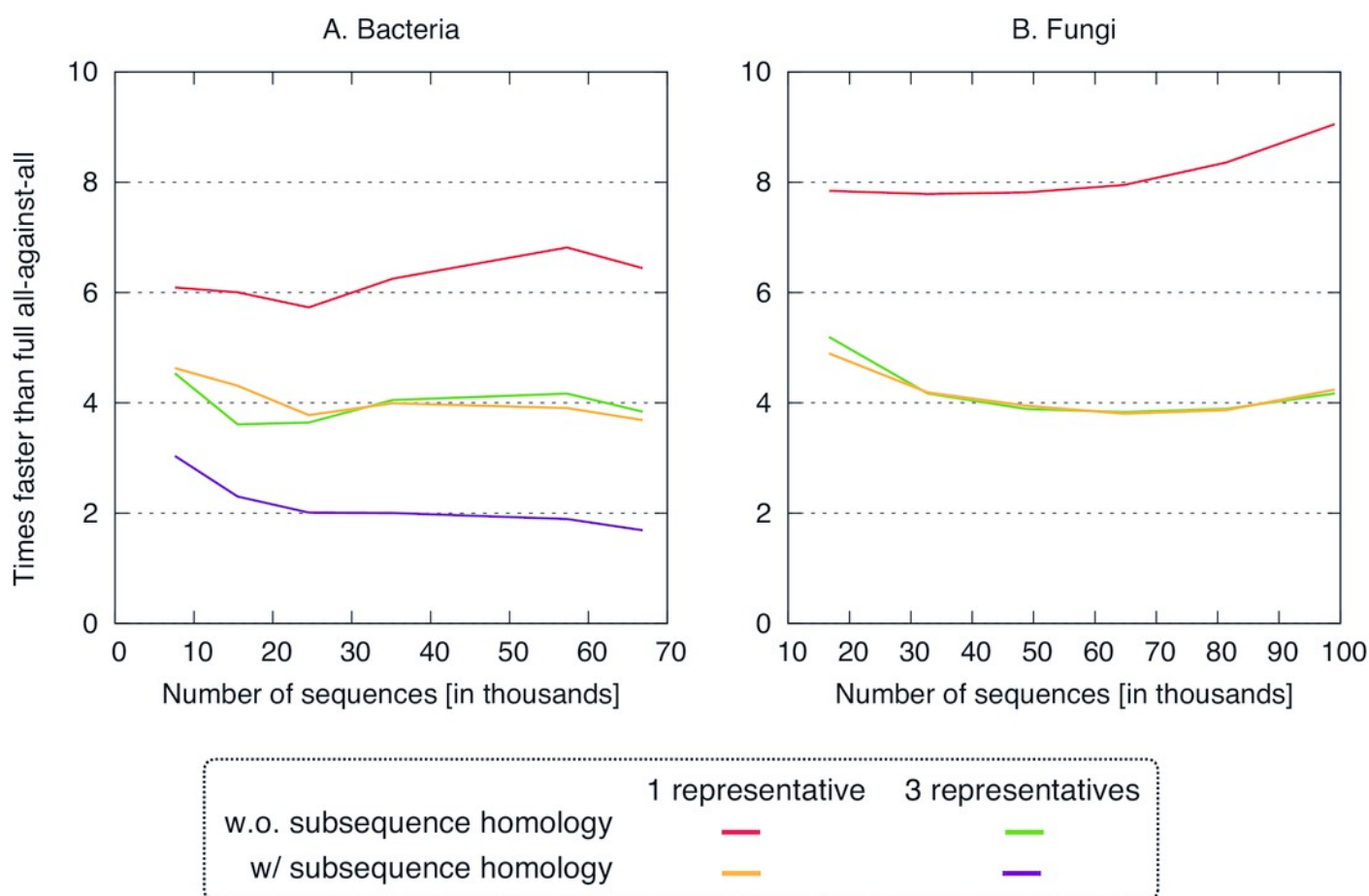



Figure 4

Figure 4: Fraction of the pairs missed when using the proposed algorithm.

The new approach misses 0.4-6% of the pairs from the full all-against-all in its simple variant (1 and 3 representatives), and 0.01-0.3% taking into account sub-sequence homology.

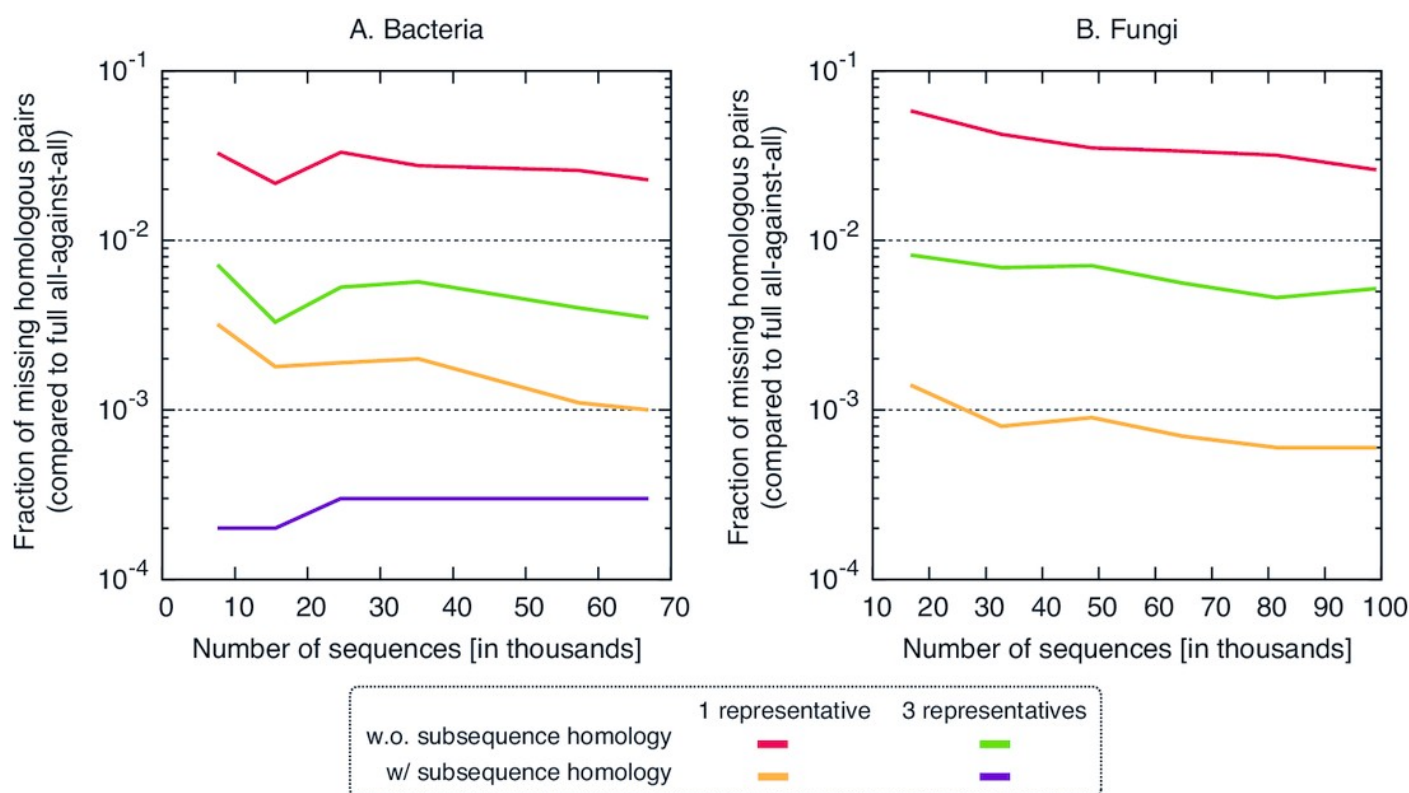



Figure 5

Figure 5: Distribution of alignments scores of missing pairs compared with all pairs ("background").

With a single representative, the distribution is heavily skewed toward low-scoring pairs.

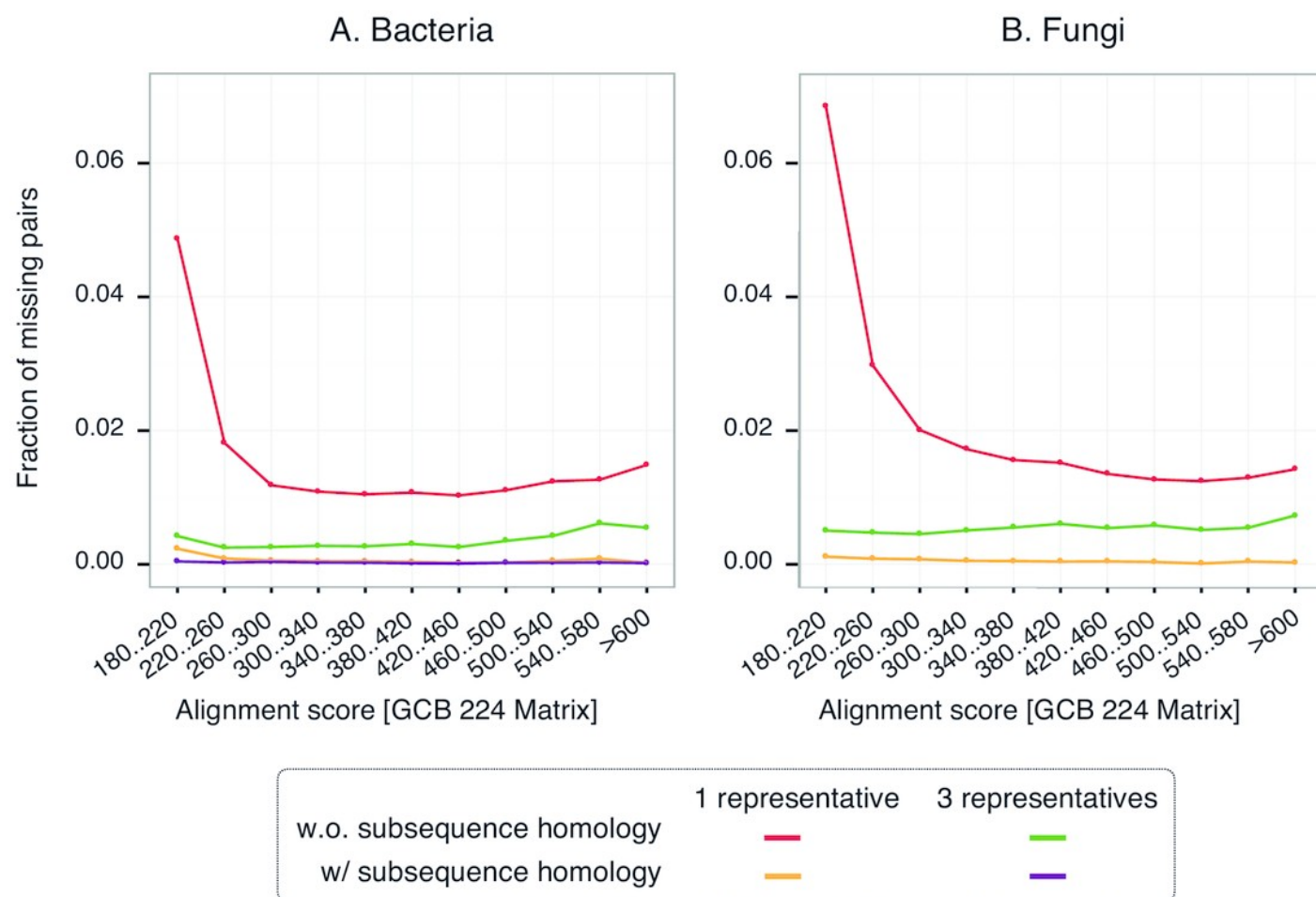



Figure 6

Figure 6: Runtime of typical K-*mer* approaches.

kClust and UCLUST are several orders of magnitude faster than full all-against-all.

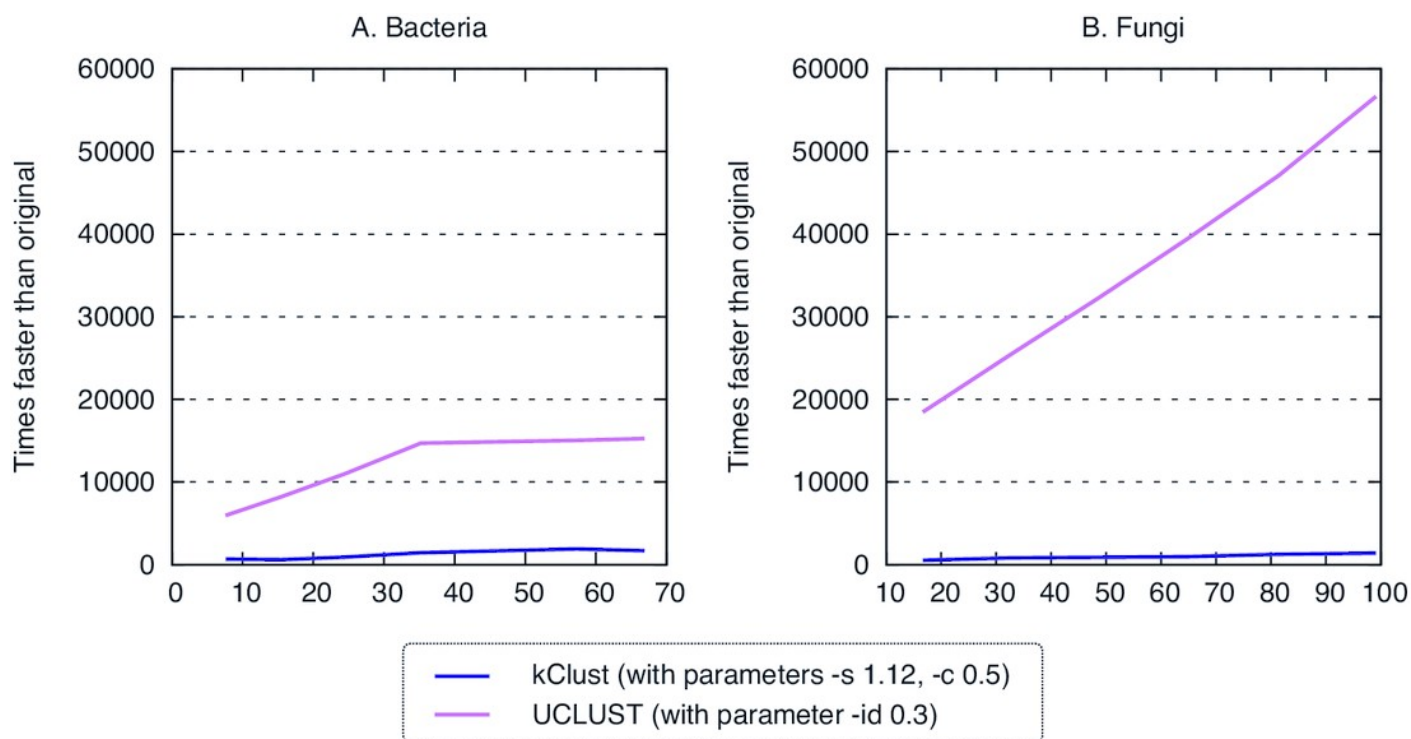



Figure 7

Fig. 7: Recall of typical K-*mer* approaches.

kClust and UCLUST only recover 3-14% of the homologous pairs identified by the full all-against-all procedure, even when they are run with sensitive parameter settings (see Materials and Methods for details).