Complet+: a computationally scalable method to improve completeness of large-scale protein sequence clustering

Rachel Nguyen [1] , **Bahrad Sokhansanj** [1] , **Robi Polikar** [2] , **Gail L Rosen** [Corresp. 1]

[1] Drexel University, Philadelphia, United States

[2] Rowan University, Glassboro, NJ, United States

Corresponding Author: Gail L Rosen
Email address: empr3ss@gmail.com

A major challenge for clustering algorithms is to balance the trade-off between homogeneity, i.e., the degree to which an individual cluster includes only related sequences, and completeness, the degree to which related sequences are broken up into multiple clusters. Most algorithms are conservative in grouping sequences with other sequences. Remote homologs may fail to be clustered together and instead form unnecessarily distinct clusters. The resulting clusters have high homogeneity but completeness that is too low. We propose Complet+, a computationally scalable post-processing method to increase the completeness of clusters without an undue cost in homogeneity. Complet+ proves to effectively merge closely-related clusters of protein that have verified structural relationships in the SCOPe classification scheme, improving the completeness of clustering results at little cost to homogeneity. Applying Complet+ to clusters obtained using MMseqs2's clusterupdate achieves an increased V-measure of 0.09 and 0.05 at the SCOPe superfamily and family levels, respectively. Complet+ also creates more biologically representative clusters, as shown by a substantial increase in Adjusted Mutual Information (AMI) and Adjusted Rand Index (ARI) metrics when comparing predicted clusters to biological classifications. Complet+ similarly improves clustering metrics when applied to other methods, such as CD-HIT and linclust. Finally, we show that Complet+ runtime scales linearly with respect to the number of clusters being post-processed on a COG dataset of over 3 million sequences. Code and supplementary information is available on Github: https://github.com/EESI/Complet-Plus.

Complet+: A Computationally Scalable Method to Improve Completeness of Large-Scale Protein Sequence Clustering

Rachel Nguyen[1], Bahrad Sokhansanj[1], Robi Polikar[2], and Gail L. Rosen[1]

[1]**Ecological and Evolutionary Signal-processing & Informatics Lab Drexel University, Philadelphia, PA, USA**
[2]**Signal Processing and Pattern Recognition Laboratory, Rowan University, Glassboro, NJ, USA**

Corresponding author:
Gail Rosen[1]

Email address: glr26@drexel.edu

ABSTRACT

A major challenge for clustering algorithms is to balance the trade-off between homogeneity, i.e., the degree to which an individual cluster includes only related sequences, and completeness, the degree to which related sequences are broken up into multiple clusters. Most algorithms are conservative in grouping sequences with other sequences. Remote homologs may fail to be clustered together and instead form unnecessarily distinct clusters. The resulting clusters have high homogeneity but completeness that is too low.

We propose Complet+, a computationally scalable post-processing method to increase the completeness of clusters without an undue cost in homogeneity. Complet+ proves to effectively merge closely-related clusters of protein that have verified structural relationships in the SCOPe classification scheme, improving the completeness of clustering results at little cost to homogeneity. Applying Complet+ to clusters obtained using MMseqs2's `clusterupdate` achieves an increased V-measure of 0.09 and 0.05 at the SCOPe superfamily and family levels, respectively. Complet+ also creates more biologically representative clusters, as shown by a substantial increase in Adjusted Mutual Information (AMI) and Adjusted Rand Index (ARI) metrics when comparing predicted clusters to biological classifications. Complet+ similarly improves clustering metrics when applied to other methods, such as CD-HIT and `linclust`. Finally, we show that Complet+ runtime scales linearly with respect to the number of clusters being post-processed on a COG dataset of over 3 million sequences. Code and supplementary information is available on Github: `https://github.com/EESI/Complet-Plus`.

1 INTRODUCTION

A critical step in many bioinformatics pipelines is clustering similar nucleotide (DNA or RNA) and protein sequences, such as clustering marker genes that may represent similar taxa or clustering peptides that may have structural homology or similar functions. In biological applications, the number of sequences that must be clustered is extremely large. Annotated protein sequence databases contain hundreds of thousands (of experimentally validated) and tens to hundreds of millions (of predicted) sequences. Metagenome sequencing experiments routinely generate hundreds of millions of reads. As a result, bioinformatics demands scalable clustering methods. Clustering methods generally have a trade-off of increased sensitivity with slower speed, making the protein clustering a challenging problem. Protein clusters are groups of similar (homologous) proteins that most likely share the same or similar function. Clustering procedures must allow compression of information in comparison to the non-clustered representation and may have different resolution levels (e.g. phyla/genus for taxa or superfamily/family for protein).

One of the key challenges in many biological applications is to generate clusters which correspond to the "true" groupings of sequences in the application. For example, if the task is to identify the

phyla present in a data set of DNA sequences, ideally clusters should correspond to groups of phyla. If the task is to identify protein families or superfamilies—the focus of this paper—then ideally clusters should correspond to protein families. This would then permit clusters to be used for the labeling of unknown sequences. For example, if an unknown protein clusters with proteins from a given family, then the clustering result may be used to predict that the unknown protein is a member of that family. As described above, however, protein sequence data sets are very large, and as a result the sensitivity-speed trade-off requires lowering sensitivity, resulting in relatively "tight" clusters: clusters that are made up of biologically similar proteins, but with groups of biologically similar proteins, such as a family or superfamily, split up into multiple clusters. In other words, the resulting clusters have relatively high *homogeneity* but relatively low *completeness*.

Optimizing homogeneity over completeness poses a barrier for certain important applications – two examples of which are described here. *First*, a common use case for clustering is to label otherwise unidentified sequences, such as hypothetical ORFs identified from a metagenomic sequencing data set. Hypothetical ORFs that are most proximate to a cluster, or cluster together with known proteins, can be labeled according to the majority –otherwise, their cluster membership can be determined with consensus. However, clustering methods with low completeness may generate lots of clusters with only one member ("singletons") or with only two or a few members—which may not have reliable consensus annotation. *Second*, clusters can be used to identify and organize groups of sequence homologs, for example to predict functional categories of proteins or groups of proteins with common evolutionary histories. Low-completeness clusters fail in this task as well, particularly as homologs become more remote or when the levels of organization become higher, for example in identifying the members of the same superfamily as opposed to family of protein sequences. Paccanaro et al. (2006); Bernardes et al. (2015) Increasing completeness would require either a slower (more sensitive) clustering method—not feasible for most biological sequence applications—or developing a method to robustly and reliably combine clusters in a way that increases completeness without excess cost in homogeneity. Notably, any decrease in homogeneity means a risk of "false positives", where a cluster includes members that are biologically dissimilar to each other.

In this paper, we introduce Complet+, a novel method to increase the completeness of clusters obtained using large-scale biological sequence clustering methods. Complet+ addresses a key problem with large-scale clustering methods, such as mmSeqs2 clustering and CD-HIT. Because large-scale clustering tools generally use some kind of iterative or heuristic method, like iterative greedy clustering, they will generally not generate a measure of identity between representative sequences of clusters. Complet+ utilizes the fast search capabilities of MMSeqs2 to identify reciprocal hits between the representative sequences, which may be used to reform clusters and 1) reduce the number of singletons and small clusters and 2) create larger clusters that better represent superfamilies composed of remote homologs, proteins that share a very distant evolutionary origin. This paper begins with a background of the relevant methods and then describes the Complet+ algorithm and the computational experiments used to demonstrate and validate Complet+ on extended Structural Classification of Proteins (SCOPe) families and superfamilies of proteins with structural homology Chandonia et al. (2021). We show that Complet+ can effectively increase completeness with a small or negligible cost to homogeneity on clusters generated using MMSeqs2's default cascade clustering method, singe-step clustering, incremental clustering with MMSeqs2's `clusterupdate` module Steinegger and Söding (2017), `linclust` (a linear-time clustering method) Steinegger and Söding (2018), and the popular clustering tool for very large-size sequence datasets, CD-HIT Li and Godzik (2006). We also show that Complet+ can result in clusters that are far more representative of the true SCOPe family and, in particular, superfamily organization. Finally, we demonstrate that Complet+ is scalable, with linear-time performance with respect to the number of clusters using the same clustering method, and with a manageable cost when used on methods that generate highly similar clusters.

2 BACKGROUND

2.1 Sequence Clustering

The large scale and complexity of nucleotide and protein sequence data have motivated the development of clustering methods specifically to address these applications. CD-HIT was one of the first such large-scale clustering methods for DNA and protein sequences Li and Godzik (2006). CD-HIT's essential innovation was to estimate similarity between sequences using k-mers (or short words) rather than performing

costly $\mathcal{O}(mn)$ sequence alignment, via dynamic programming, or a heuristically-sped up version like BLAST. CD-HIT can be used for a wide range of applications due to its versatility, but one particular application—16S rRNA clustering for de novo taxonomic grouping (a.k.a open-reference OTU picking)— immediately became an important application due to the volume of such data being generated Chen et al. (2013). Competing clustering methods soon emerged, each taking a different approach to speed-accuracy tradeoffs, such as UCLUST Edgar (2010). Historically, however, these tradeoffs have been evaluated to address 16S rRNA sequence clustering, which has particular issues with accuracy that do not apply to protein clustering Nguyen et al. (2016); Schloss and McMahon (2021).

Protein sequence clustering methods are similarly being investigated with an even wider gap between speed and accuracy Hauser et al. (2016). There are two categories of clustering algorithms: alignment-based and alignment-free. The traditional alignment-based methods, CD-HIT Li and Godzik (2006), UCLUST Edgar (2010), and BlastClust for Biotechnology Information (NCBI) can also be used for proteins in addition to 16S rRNA. Futher developments of Markov Clustering (MCL) techniques were innovated Enright et al. (2002); Wong and Ragan (2008). However MCL techniques, as well as other alignment-based techniques, are computationally expensive. Alignment-free based techniques are emerging, with MMSeqs2 Steinegger and Söding (2017) being the most popular, and with emerging techniques in deep learning and embeddings Karim et al. (2020). MMSeqs2 clustering is widely used in a range of applications, including clustering known and unknown genes in metagenomes Vanni et al. (2022).

2.1.1 MMseqs2

Many-versus-Many sequence searching (MMseqs2) is a software suite that offers a wide array of tools for sequence alignment Steinegger and Söding (2017) and clustering Steinegger and Söding (2018); Hauser et al. (2016) of both nucleotide and protein sequences. It has high sensitivity and is efficient in both computation time and hardware resources, as it is optimized for multi-threaded use. It is also highly scalable and excels at working with large datasets. In this study, we examine MMseqs2's sequence alignment and clustering tools.

Sequence alignment Steinegger and Söding (2017) is performed using MMseqs2's `search` module, and is comprised of three steps: a k-mer match step, a vectorized ungapped alignment, and a gapped (also known as Smith-Waterman) alignment. The k-mer match step drastically reduces the number of Smith-Waterman alignments, and is a main contributor to the `search` module's high speed.

A search is characterized by many criteria and parameters, one of the more notable ones being its sensitivity. A search run with a low sensitivity setting will be fast, but it will find less hits than a search using a high sensitivity setting. The `search` module also offers the ability to restrict the output to alignments that are within a specified significance threshold (also known as expectation-value, or e-value).

Two major modules within MMseqs2 for clustering are: `linclust`, a linear method, and `cluster`, a cascaded method that begins with `linclust` before performing additional prefiltering, alignment, and clustering. Both clustering modules are unsupervised and their results are independent of the order sequences are presented in. (MMseqs2 further offers the option of simple clustering in a single step without cascading, but that is not its default operation.)

MMseqs2's implementation of linear clustering is the `linclust` module Steinegger and Söding (2018). The computational runtime of `linclust` scales linearly with the number of input sequences. The `linclust` algorithm begins by grouping sequences that share a k-mer, selecting the longest of sequence of each group as the center (or representative sequence). Every sequence is then compared (using Smith-Waterman alignment) to every representative it shares a k-mer with, and if it passes the clustering criteria it is recruited into the cluster.

The cascaded clustering module is the default setting of the `cluster` module in MMseqs2 Steinegger and Söding (2018). The cascaded clustering method first runs the `linclust` module to produce an initial clustering. It then performs a prefiltering step followed by a Smith-Waterman alignment of the sequence pairs that passed. Next, an initial clustering with low sensitivity and high significance threshold is done, which results in a quick clustering that only yields matches with high sequence identity. It then repeats the process beginning at the prefiltering step, instead using the first step's cluster representative sequences rather than all sequences. It also uses a higher sensitivity than the first step. The process is repeated a third time using the sensitivity specified by the user, and the results are finally merged. Cascaded clustering can be performed with as many as seven steps, but three is the default.

The `cluster` module's sensitivity setting is similar to search's sensitivity setting: a low sensitivity clustering will be faster than one at the highest sensitivity, but the result will be comprised of many

155 more clusters due to finding less hits. The `cluster-reassign` option will recompute the clusters'
156 representative sequences at each clustering step. Otherwise, `cluster` will use the representative
157 sequence initially chosen for each cluster at each step.
158   The `cluster` module's `single-step-clustering` option changes the workflow to perform
159 the prefiltering, alignment, and clustering of the sequences using the user's specified criteria at the first
160 and single step, as opposed to gradually over the course of three steps like `cluster`.

161 2.2 Incremental Learning and the Cluster Updating Workflow in mmSeqs2

162 Incremental learning is a developing approach to handling the rapid growth of massive amounts of
163 biological sequence data, particularly in the era of next-generation sequencing. Classically, a database
164 needs to be *de novo* clustered or a supervised machine learning classifier needs to be fully retrained each
165 time a sequence database is updated with new examples. An incremental approach can instead learn
166 fundamental features and classes of the data continuously as new examples are added Zhao et al. (2020).
167 Both supervised and semi-supervised methods have been proposed for classifying sequences Ozdogan
168 et al. (2021); Zhao et al. (2020); Dash et al. (2021). One incrementalization approach is to reduce the
169 time required for sequence alignment by saving information and running only on new increments Dash
170 et al. (2021). Another is to incrementally update gene/genome classifiers, such as NBC++ Zhao et al.
171 (2020) and Struo2 Youngblut and Ley (2021). Our group has begun to explore incrementalization methods
172 for unsupervised methods that update the cluster representatives, and semi-supervised methods that rely
173 on clustering information in combination with learning classifiers for both taxa and protein sequence
174 classification Halac et al. (2021); Ozdogan et al. (2021).
175   MMseqs2 Steinegger and Söding (2018) is one of the most commonly used software suites for
176 protein sequences in academic and commercial applications. It is the only one that offers incremental
177 cluster updating. The `clusterupdate` module allows the user to update a previous clustering result
178 following the addition or removal of sequences. When updating the clustering result, the module keeps
179 the sequences' identifiers consistent within the original clustering's databases. It also shares many of the
180 settings the cluster module offers, most notably the sensitivity setting. The `clusterupdate` module
181 first compares the new sequences to existing ones with a supervised process. Then, when sequences which
182 are not similar enough to any of the existing clusters are added, they are compared to each other in an
183 unsupervised process, with some becoming representatives of new clusters.

184 3 METHODS

185 All tests were performed utilizing a single CPU thread and 16 GB of memory.

186 3.1 Protein Data Sets

187 3.1.1 Structural Classification of Proteins (SCOP)

188 Developed by MRC Laboratory of Molecular Biology, the SCOP database classifies proteins first by their
189 secondary structure, followed by their evolutionary relationships primarily through manual curation until
190 work on it ceased June 2009 Hubbard et al. (1997). Its development was continued with the release of
191 SCOP - extended (SCOPe) database Fox et al. (2013) in March 2012 by Berkeley Lab and UC Berkeley,
192 who continue to expand it with a combination of manual curation and automated methods. The data itself
193 are the genetic domain sequences of proteins within PDB SEQRES records.
194   As of version 2.08, SCOPe contains 302,566 protein domain sequences in total. The topmost level
195 of classification is sequence class, followed by fold, superfamily, and lastly family. SCOPe 2.08 is
196 encompassed by 7 classes, 1257 folds, 2067 superfamilies, and 5084 families Chandonia et al. (2021). For
197 this study, we only evaluate the clustering results on the superfamily and family levels of classification.

198 3.1.2 Clusters of Orthologus Genes (COG)

199 The COG database Galperin et al. (2019) groups protein functional domains into COG's according to
200 their orthologous relationships. Each COG thus represents a general function. For example, glutamyl-
201 or glutaminyl-tRNA synthetase proteins are classified under COG0008. Each of these COGs are further
202 defined by one or more functional groups, there being 26 functional groups total. The current release as of
203 May 12, 2022 includes 3,456,089 functional domains encompassed by 4877 COGs Galperin et al. (2021).

204 3.1.3 Swiss-Prot COG

205 The Swiss-Prot database Bairoch and Apweiler (1999) is a subset of the Uniprot. The database is
206 composed of manually-annotated records with information extracted from literature and curator-evaluated

computational analysis. We filter the COG annotations using only these manually-annotated proteins to obtain biologically-verified COGs. The Swiss-Prot COG subset contains 122,202 functional domain sequences encompassed by 3,858 COGs.

3.2 The Complet+ Algorithm

The Complet+ algorithm builds on unsupervised clustering results by aligning cluster representatives (via MMSeqs2 search) and merging clusters that are reciprocal hits within a particular e-value. In the results, we show that Complet+ can be used on any base algorithm, thus it implements "incremental learning" without needing all previous data (just cluster representatives and a mapping of sequences to clusters), and it runs in less time than rerunning a more sensitive clustering algorithm from scratch. Also, currently, there are no other postprocessing algorithms that improve completeness without having to be rerun, from scratch, on all the data. Therefore, Complet+ fills a niche in optimizing clustering for completeness in an incremental learning manner.

Complet+ takes any 1) FASTA file of representative (or all) sequences and 2) clustering algorithm results as input after they have been converted to a particular TSV format, which is a simple two-column data file of sequence IDs and the IDs of their cluster representatives. For example, MMseqs2 results are obtained from MMseqs2's `createtsv` command, but Complet+ is compatible with any clustering data file regardless of its source, provided it is converted into the same format. The FASTA file of the sequences only needs to contain the sequences of the cluster representatives, but if there are additional sequences, the algorithm extracts the relevant ones. Representative sequence extraction is performed through the use of grep, a command-line tool used for searching patterns within files. The representative sequences, as described in the cluster data file's first column, are all extracted to a new FASTA file. The clustering data file is used again in the relabeling step (Fig. 1).

With this FASTA file of representative sequences, Complet+ then performs a pairwise sequence alignment between all representatives by using the MMseqs2 search function, with the pairwise combinations as query and target arguments. We found that the homogeneity-completeness was optimized at the highest sensitivity option, s=7.5, of the MMseqs2 `search` function. As shown in the Cascade-C+ results in Results and Discussion Section, a Complet+ `search` sensitivity of 7.5 (s=7.5) increased the Default completeness by 5.99% (while lowering homogeneity by 0.05%), while a Complet+ `search` sensitivity of 5.7 (s=5.7) only increased the Default completeness by 3.63% (while lowering homogeneity by 0.02%). The other settings, such as e-value threshold and query-coverage, were left at the default values of 0.001 and 0.8, respectively. Also, Complet+ can even improve upon Cascade Clustering and Connected Component (CC) clustering with the max sensitivity of 7.5. The algorithm employed by Complet+ is similar to connected component clustering (cluster mode 1 in MMSeqs2), but has an even looser constraint to improve completeness. The looser constraint is that instead of connecting nodes that are best reciprocal hits by raw score, Complet+ looks for all reciprocal hits that fall within a user-defined e-value. From the reciprocal hits, we are able to deduce which clusters are similar to each other, in order to merge them via the following described process.

After the pairwise alignments of the cluster representatives are obtained, the alignment results are sorted from lowest e-value to highest and then filtered, retaining only sequences that are "reciprocal hits" within the specified e-value to the algorithm. Reciprocal hits are pairs of sequences where one is in the top hits of the other and vice-versa when queried against all sequences (in this case, all the cluster representatives). Alignment results that are "self-hits" (where the query and target sequences are the same sequence) are filtered out. Fig. 2 depicts how only the reciprocal hits are retained following the alignment filtering step. It is important to note that the reciprocal hits need not be the "reciprocal best hits", or best-scoring hits for the sequences. A sequence can and likely will have multiple sequences to which it is reciprocal.

We observed that the reciprocal hits obtained through this method formed clusters of their own. A given representative sequence would usually have between two and six reciprocal hits, following the sequence alignment. Those reciprocal hits would tend to only be reciprocal hits to each other. Fig. 3 visualizes these reciprocal-hit results and the clustering observed. As evident in the visualization, these reciprocal hits form numerous isolated groups akin to clusters. We relabel the members of these clusters with the ID of one of the members in order to merge them.

Beginning with the first representative sequence in the file, all subsequent representative sequences with which a sequence is reciprocal are relabeled as the original representative sequence. Complet+ then

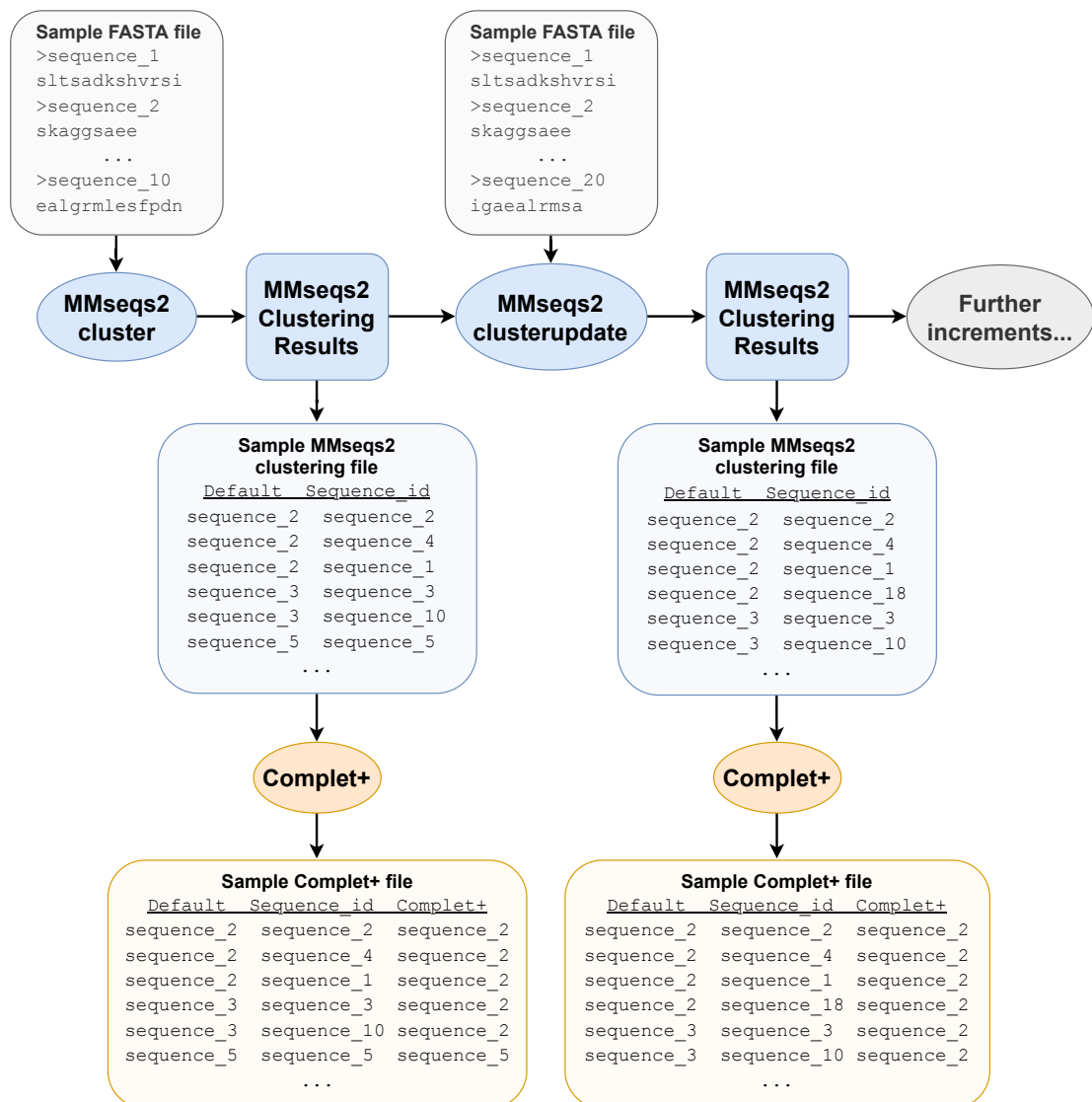



Figure 1. Example pipeline depicting Complet+ usage. Not all files required and produced by the MMseqs2 tools and Complet+ are depicted, merely the most prominent ones, for clarity. The pipeline depicted only includes one clusterupdate module use following the initial clustering with the cluster module, but the tests discussed later feature more successive increments with clusterupdate.

261 iterates through each of those reciprocal hits' own reciprocal hits, which are also relabeled as that first
262 sequence. Complet+ traverses through the data in this recursive fashion, ignoring representative sequences
263 that have already been passed. All of the members of that original representative sequence are hence
264 identified and relabeled. Complet+ repeats this process for every representative sequence in the file,
265 ignoring those that have already been seen. This process results in a list of new representative sequences
266 which each former representative sequence and its cluster members are relabeled to. The old cluster ID
267 and Complet+ cluster ID, corresponding to each sequence ID are then saved into the Complet+ output,
268 seen in Fig. 1.

269 The algorithm then passes through the data file of the MMseqs2 clustering results. If a sequence's
270 representative is on the aforementioned relabeling list, that sequence is relabeled with the new representa-
271 tive sequence ID. The resulting clustering data file contains the data from the original clustering file, in
272 addition to the new cluster representative of each sequence (Table. 1).

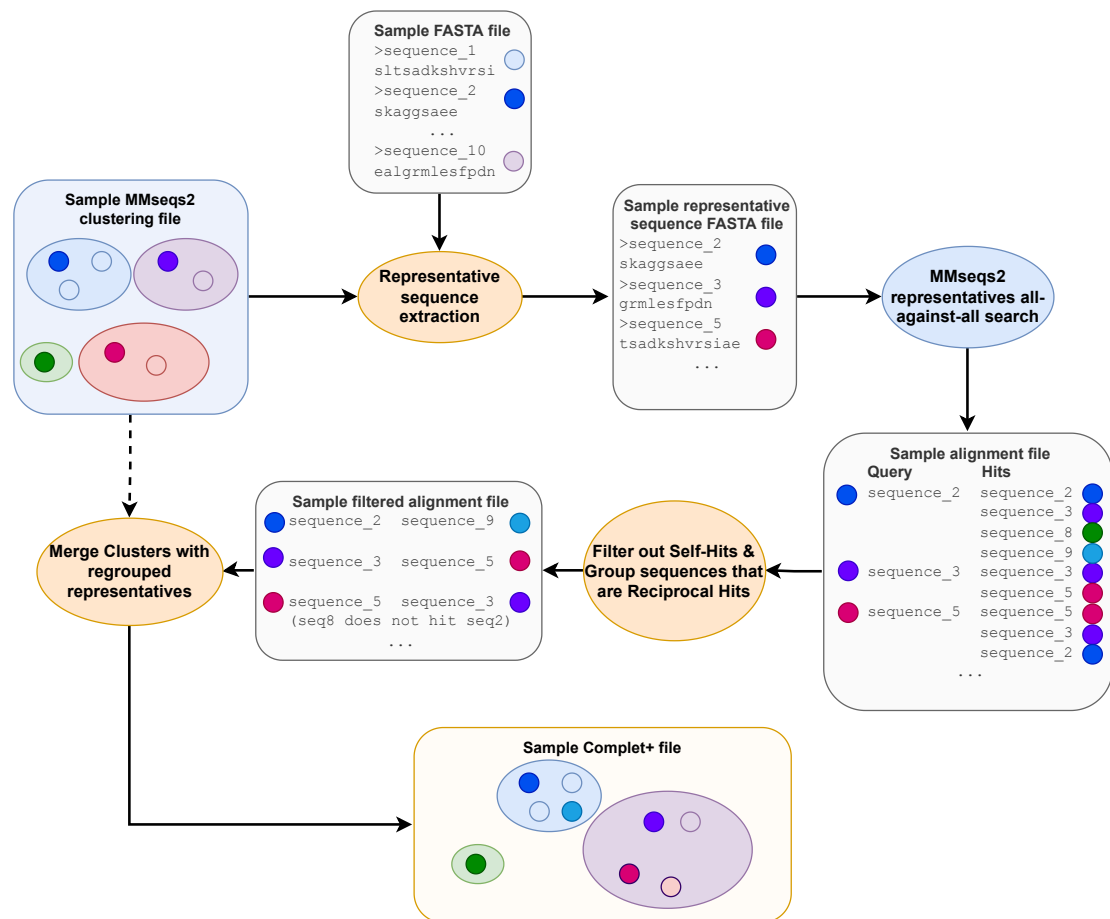



Figure 2. The Complet+ algorithm. The alignment files contain additional data columns, including the one containing each alignment's e-value (not depicted) that is used in the alignment sorting and filtering step.

3.3 Clustering Metrics

To evaluate the performance of both MMseqs2 and our post-processing algorithm, we used Scikit-learn's cluster metrics library Pedregosa et al. (2011) to calculate homogeneity, completeness, adjusted mutual information score (AMI), and adjusted Rand index (ARI).

Table 1. The output of Complet+. The file retains the representative sequence ID for each sequence's old cluster label, allowing simple comparison between the old and new labels.

Old cluster ID	Sequence ID	Complet+ cluster ID
d1u9ca_	d1u9ca_	d1u9ca_
d1u9ca_	d1pv2c_	d1u9ca_
d1u9ca_	d1pv2d_	d1u9ca_
d1n57a_	d1n57a_	d1u9ca_
d1n57a_	d1izya_	d1u9ca_
d1n57a_	d1izza_	d1u9ca_

3.3.1 Homogeneity and Completeness

Homogeneity and completeness are two metrics often used together – and can also be combined into a V-measure – to evaluate clustering results Rosenberg and Hirschberg (2007). Homogeneity is a measure of a clustering purity: a clustering result has perfect homogeneity (of 1) if all members of any given cluster are truly of the same class. Homogeneity score is penalized (reduced) if its clusters contain members that are of different classes. A clustering result has perfect completeness (of 1) if all members of each class are

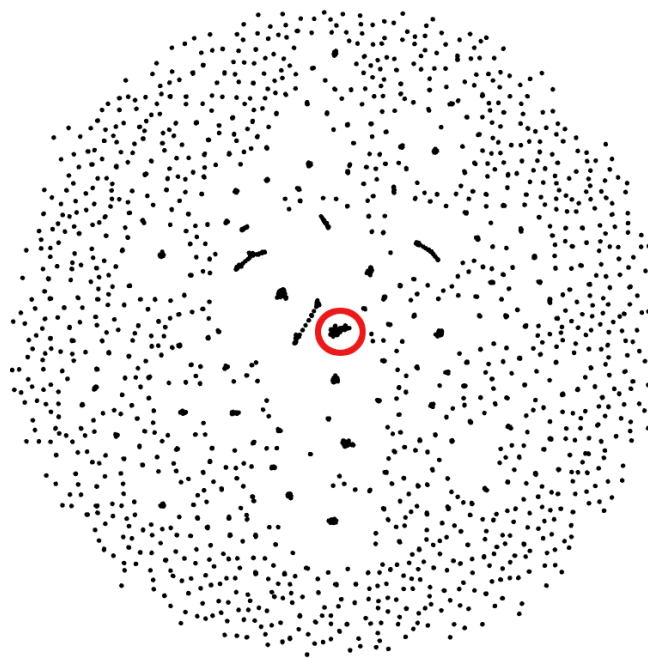



Figure 3. Gephi visualization of the MMseqs2 search of the representative sequences. Depicted are the reciprocal hits resulting from the sequence alignment. Each dot represents a representative sequence. Dots in close proximity indicate a reciprocal hit. Groups of close reciprocal hits are effectively clusters. Complet+ merges these clusters to improve completeness. Circled in red is one such cluster. Every representative sequence within this circle is a reciprocal hit to at least one other representative sequence in the circle. Complet+ merges these representative sequences, resulting in a single clusterEvery single sequence in the image has a reciprocal hit; a dot that appears to be a single sequence is actually two or more dots on top of each other.

283 within the same cluster. Completeness score is penalized (reduced) if any class members are split amongst
284 multiple clusters. Finally, V-measure is the harmonic mean between homogeneity and completeness and
285 is also known as the normalized mutual information metric Rosenberg and Hirschberg (2007); Pedregosa
286 et al. (2011).

287 **3.3.2 Adjusted Mutual Information (AMI) and Adjusted Rand Index (ARI)**

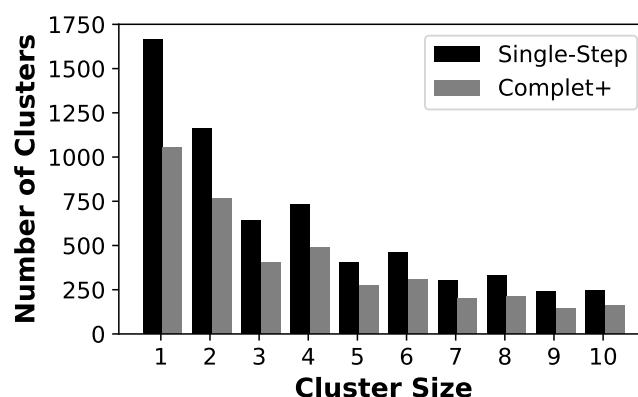



Figure 4. Number of clusters having one (singletons) through ten members for the MMSeqs single-step clustering before and after applying Complet+.

288   The mutual information score (MI) is an information theoretic measure of agreement between two

clusterings. In our case, the two clusterings are the true clusters (as determined by ground-truth classes) and the predicted clusters. The AMI is obtained by correcting MI for chance Vinh et al. (2009). The Rand index (RI) is a measure of similarity between two clusterings. ARI is the RI adjusted for chance Vinh et al. (2009). While superficially similar, AMI and ARI in fact reflect different relationships between predicted clusters and ground-truth classifications depending on whether the clustering solutions are balanced or imbalanced. A balanced clustering solution is one of equally-sized clusters, whereas an imbalanced solution's cluster sizes vary greatly. An imbalanced solution is more likely to present clusters that are pure, due to likely having more small clusters than a balanced solution. Specifically, AMI tends to give higher scores to (or is biased towards) unbalanced clustering solutions, while ARI is biased towards those that are balanced Romano et al. (2015).

4 RESULTS AND DISCUSSION

A primary goal of Complet+ is to serve as a versatile approach that can be used with any clustering algorithm. Therefore, we evaluated Complet+ on a variety of initial clustering algorithms. We also evaluated Complet+ on both classical single-batch analysis and the incremental multi-batch analysis used for incremental learning applications: specifically, single-batch clustering tests, two 5-batch incremental learning tests, and a 50-batch incremental learning test.

In single-batch clustering, Complet+ is run after the SCOPe dataset is clustered with a single cluster call (using variety of methods provided by mmSeqs2, as well as CD-HIT). Complet+ was then run on the resulting clustering output. In addition, two batch partitioning test data sets were created to help further evaluate the performance and scalability of the method as described below: 1) for each of the 5 batches, ∼20% of the new classes (previously unseen by the clustering method) were added to each batch, and 2) each of the sequences in the 5 batches were randomly chosen. Both tests result in five folds of approximately equal number of proteins. To evaluate incremental learning applications, we create a simulated scenario by inputting the first batch to MMseqs2's default cascaded `cluster` module, then sequentially adding each of the remaining four groups to the database using MMseqs2's `clusterupdate` module. Complet+ is then run on he MMseqs2's clustering results from each of these five steps. Using a cross-validation approach, this is repeated an additional four times, for a total of five folds. For the "new classes" and "random" partitioned data, the clustering metrics were averaged (and standard deviation was calculated) across the five folds. Clustering metrics were calculated between predicted labels (the clustering result) and true labels (SCOPe family/superfamily).

4.1 Complet+ Improves Clustering Metrics

4.1.1 Applying Complet+ to MMseqs2 clustering and CD-HIT

We first show that Complet+ significantly reduces the number of singletons and small clusters that are formed. Fig. 4 shows the reduction in the number of small clusters when applying Complet+ to the results of mmSeqs' single-step clustering on the SCOPe data set. There is a significant reduction in the number of singletons, and consistent reductions of up to 50% in the numbers of clusters with few representatives. While not shown on the graph, Complet+ also generates larger clusters than default methods.

Overall, CD-HIT produces numerous clusters, clustering the 303,000 SCOPe sequences down to about 204,000 clusters. The proliferation of clusters is evident in CD-HIT's low completeness score shown in Fig. 5. MMseqs2 generated fewer clusters than CD-HIT, its `linclust` module producing about 30,000 clusters, and the cascaded clustering tests generated between 9,000 and 10,000 clusters. These results are shown in Fig. 5 a) and Fig. 5 b). All default results had excellent homogeneity scores as well, at or above 0.95.

Fig. 5 a) and Fig. 5 b) show the homogeneity/completeness, AMI, and ARI for the single-batch tests using MMSeqs2 and CD-HIT. For all tests, Complet+ was able to substantially improve completeness without measurable loss in homogeneity at the superfamily level and minimal loss at the family levels of classification. The composite of homogeneity and completeness, also known as the V-measure (not shown in the graph but but can be calculated from the homogeneity and completeness) is substantially higher when applying Complet+ at the superfamily-level when averaged across all single-batch tests: 0.80 vs. 0.89. The increase in V-measure is less pronounced but still favorable at the family-level: 0.86 vs. 0.91. As indicated by Fig. 5 a) and Fig. 5 b), the V-measure increase is due to the homogeneity of the default Complet+ clustering remaining mostly unchanged, while completeness increasing.

The lower homogeneity of both Default and Complet+ results at the family versus superfamily level are expected. There are many more families than superfamilies. If there is a mis-clustering at the

In the first row, ●: Default Homogeneity, ●: Default Completeness, ●: Complet+ Homogeneity, ●: Complet+ Completeness.
2nd and 3rd rows: ●: Default, ●: Complet+

(a)   (b)

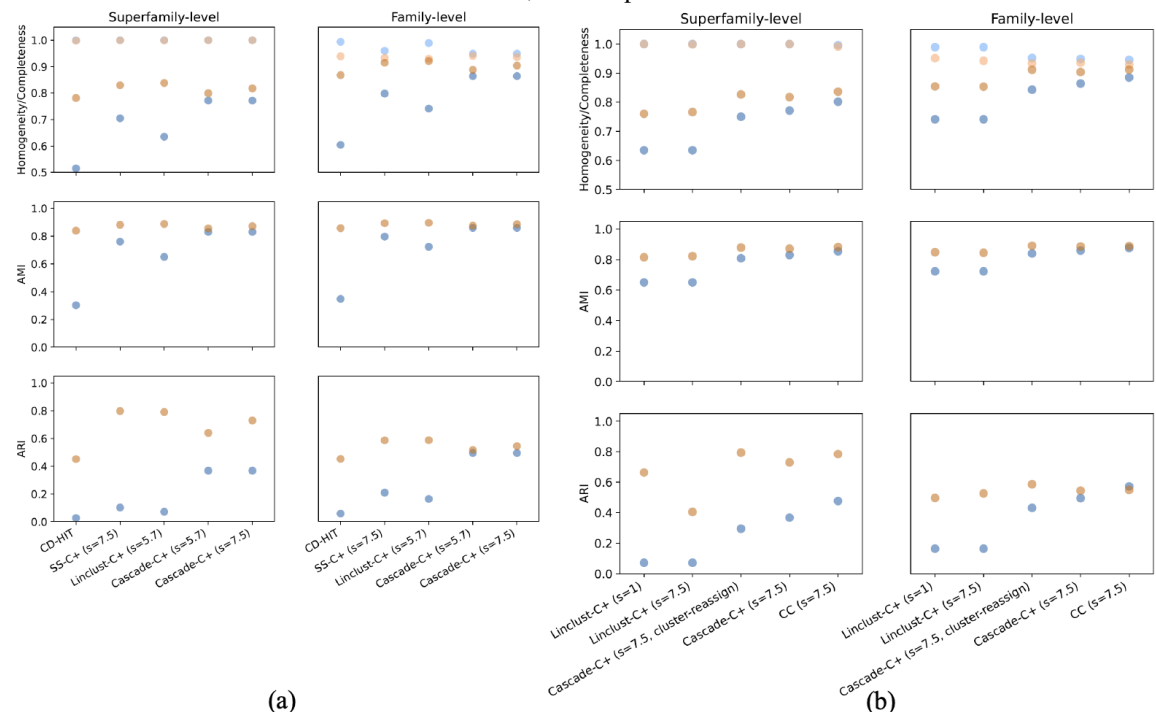



Figure 5. The homogeneity, completeness, AMI, and ARI of the single-batch tests. (a) All tests aside from "CD-HIT" use MMseqs2 the clustering module stated. The sensitivity specified in parentheses refers to the MMseqs2 `search` run by Complet+, not the clustering sensitivity, which was the default value of 4.0 where applicable. Overall, Complet+ substantially improves each test case's completeness at little expense to homogeneity. The loss in homogeneity is more notable when evaluating the clustering results on the family-level of classification, however still to a lesser degree than the increase in completeness. Complet+ also improves the AMI and ARI of each clustering to varying degrees, having a generally greater improvement. (b) The three leftmost tests are Complet+ run at minimum, default, and maximum MMseqs2 search sensitivity, each on the same Default `linclust` test results. The two following tests are identical aside from one using the cluster-reassign setting. Like the results in (a), AMI and ARI are improved with each case while completeness is also improved without significant loss of homogeneity.

superfamily-level, it will be reflected on the family level as well, but not vice versa. On the other hand, completeness is higher on the family level because there are more families than superfamilies. Both Default and Complet+ are conservative with their clustering, having a tendency to create multiple smaller clusters over large ones, which then results in lower completeness. Furthermore, since the superfamily is the larger clade above family, the score is more penalized when evaluated on the superfamily level.

As shown in Fig. 5 a) and Fig. 5 b), Complet+ also results in an increase in both AMI and ARI when compared to the Default clustering. Regardless of how low the Default AMI score was, Complet+ was able to improve it to a score above 0.8. The improvement to ARI was also notable across most test cases, where Complet+ scores were higher relative to the Default scores. For both Default and Complet+, AMI was notably higher than ARI in most cases, which was expected due to AMI's previously discussed bias towards datasets with unbalanced ground-truth classes.

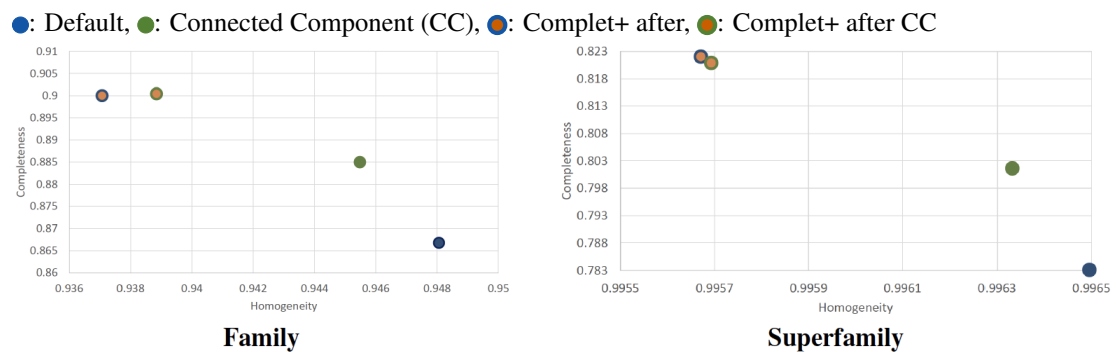



Figure 6. Homogeneity/Completeness scatterplots demonstrating Base-Algorithm + Complet+ vs. Base Algorithms (Default, Connected Component clustering both with highest sensitivity of 7.5). Complet+ can improve completeness of each algorithm by a greater amount than is lost in homogeneity. By merging clusters whose representatives are reciprocal hits (given an e-value threshold) of each other allows more clusters to be merged than stricter connected node criteria by the CC algorithm.

We can see from the plots in Fig. 5 that while there is improvement in completeness over connected clustering, the overal AMI/ARI are quite similar. In fact, for family-level, the difference in AMI is only by 1%. Upon further investigation, this small yet overall improvement can be seen for all sensitivity levels, as shown in the connected component with Complet+ supplementary graph on the Github page (https://github.com/EESI/Complet-Plus/blob/main/figures/connected_component_study/cc_fam_superfam.pdf). Examining the homogeneity/completeness in scatterplot form, Fig. 6 illustrates that improvement in completeness is still a few percentage points at the cost of a fraction of a percentage point in homogeneity for even the highest sensitivity of 7.5. Interestingly, for sensitivity of 4 (in the supplementary graphs, https://github.com/EESI/Complet-Plus/blob/main/figures/connected_component_study), the default+Complet+ is similar in performance to connected component clustering – however, improved completeness can be gained by running Complet+ on connected clustering. While these algorithms are similar, due to the incremental learning nature of Complet+, it can usually make some improvements given general similarities of the cluster representatives.

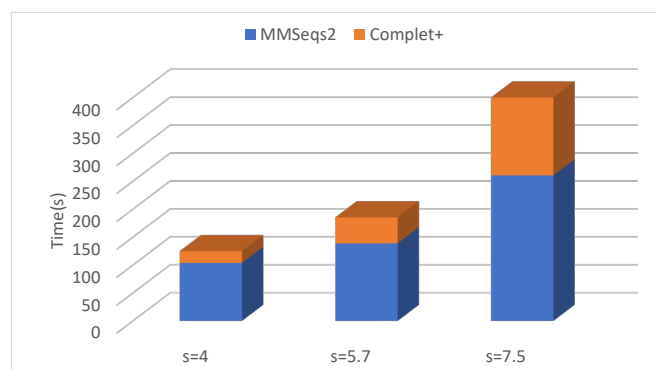
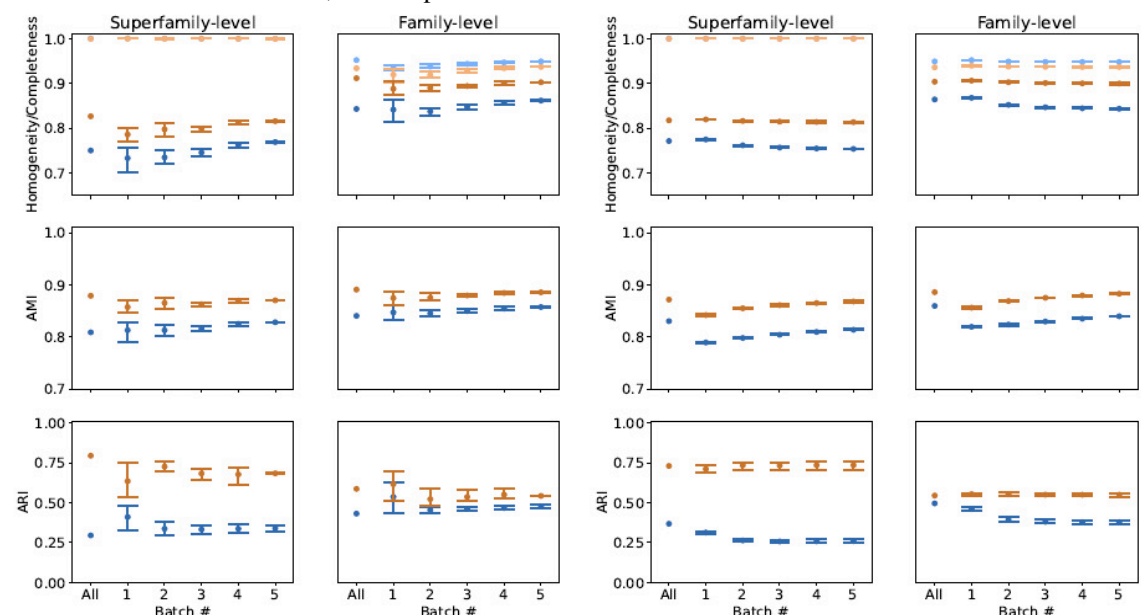



Figure 7. Algorithm runtime vs. sensitivity levels for MMSeqs2 and Complet+. Both algorithms' runtimes increase polynomially.

Finally, we can see that the improvements in completeness are notable but overall improvement is less at higher sensitivities. The runtime of both the default MMseqs2 and Complet+ significantly increase as the sensitivity increases, as shown with the connected component (CC) clustering runtimes vs. sensitivity for the two algorithms shown in Fig. 7.

In the first row, ●: Default Homogeneity, ●: Default Completeness, ●: Complet+ Homogeneity, ●: Complet+ Completeness.

2nd and 3rd rows: ●: Default, ●: Complet+

(a) Homogeneity & Completeness: "New classes" Batching

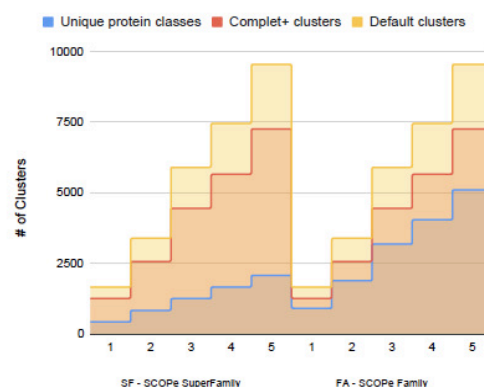
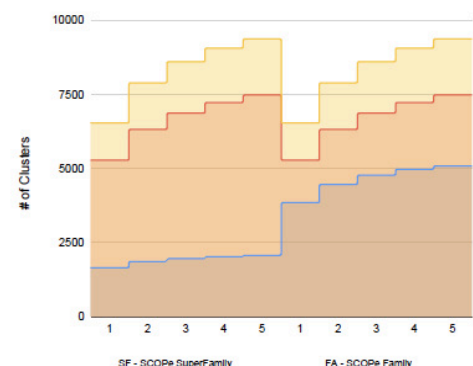
(b) "Random" Batching

(c) Number of clusters: "New classes" Batching

(d) "Random" Batching

Figure 8. (top) Homogeneity, completeness, AMI, and ARI of the superfamily- vs. family- level for both (a) "new classes" and '(b)'random" test batching partitions (for 5 batches) for MMSeqs `cluster` and `clusterupdate` followed by Complet+. The tick labeled "All" on the graphs represents clustering all sequences in one single batch. Overall, Complet+ increases MMSeqs2 completeness by substantially more than it reduces homogeneity relative to the default MMSeqs2-generated clusters. Using Complet+ results in an increased AMI and ARI at both family and super-family levels. Also, we can see that discovery of new classes yield a large variance in performance as opposed to the base algorithm obtaining most classes in the first batch. The variance is due to the number of actual families or super-families ("true" clusters). (bottom:c & d) Number of "true" and predicted clusters for default MMseqs2 and Complet+. The number of true clusters is always lower than what Default MMseqs2 finds, and Complet+ is able to reduce them 10-20% by merging proteins that belong to the same family/superfamily.

4.1.2 Batch Incremental Learning using MMseqs2 clusterupdate

The results of incremental tests are consistent with the single-batch tests. As Fig. 8 shows, applying Complet+ results in an increase in completeness significantly greater than any concomitant reduction in homogeneity for incremental clustering (i.e., V-measure is higher) as well. Notably, in Fig. 8, the "new

classes" (class-partitioned) results have a higher variance than that of the random partitioning. This is due to the difference in the data partitioning; in the case of the "new classes" test, 20% of the classes are added in each batch. In the case of the randomly-partitioned test, 20% of the data is added in each batch so the algorithm has seen the vast majority of the classes within the very first batch. MMseqs2 performs best when clustering incoming sequences if their true class is already represented in the database. This is consistent with the low variance seen in the metrics of the randomly-partitioned results. In the class-partitioned test, it is possible that some incoming classes may be similar to classes that have already been seen. Such sequences may then be clustered with existing clusters rather than being placed into new clusters. The batch that these clusters are in will vary between the tests, potentially causing increased variance in clustering metrics.

For the class-partitioned incremental tests, the metrics of Batch 1 clustering were lower than those of the single-batch clustering. With each successive batch, homogeneity/completeness, and AMI increased, and by the last batch they were nearly as high as the metrics of the single-batch test. ARI slightly decreased with successive batches indicating a bias in clustering sequences together rather than making false positives/negatives randomly (which ARI rewards). As previously explained, AMI is the better measure for ground truth clusterings that are unbalanced, and ARI is better for balanced ground truth clusterings. Both SCOPe and Swiss-Prot COG are unbalanced datasets and thus favored by AMI. With the random tests, the metrics of the MMseqs2 `clusterupdate` algorithm never reached the single-batch metrics – however, Complet+ relatively stayed the same. The homogeneity and completeness of both the "new class" and "random" tests were relatively close in value by the final batch.

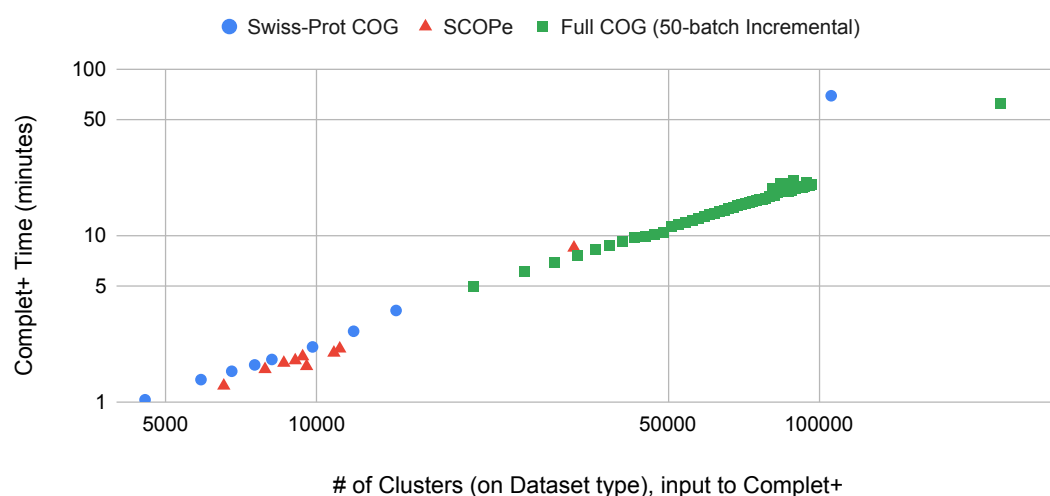



Figure 9. The Complet+ time vs. the number of clusters produced by a variety of algorithm modes in MMseqs2. Some algorithms deviate from the line on the log-log plot due to the output cluster representatives and the different relationships between them (e.g.: cascaded tends to produce more less similar representatives while `linclust` produces more similar ones). Using the same type of algorithm on the 50-batch large dataset, Complet+ scales linearly vs. the number of input clusters.

To further evaluate the potential applicability of Complet+, we also tested it on the aforementioned MMSeqs2 clustering methods and CD-HIT on the Swiss-Prot COG database. Full results are provided at our GitHub site for this paper, https://github.com/EESI/Complet-Plus. In brief, when applying Complet+, the Swiss-Prot COG (only the Swiss-Prot portion of the COG database) class-partitioned test saw similar improvements in V-measure across the board. However, single-batch and random-partitioned tests do not show the same level of improvement. While Complet+ increased the completeness of the Default clusterings, the losses in homogeneity were greater than those observed in the SCOPe tests. This result is likely due to nearly identical sequences being classified in different COGs. Accordingly, schemes like COG, which classify sequences apart despite a high degree of similarity, are not well suited for Complet+.

4.2 Complet+ is Computationally Scalable

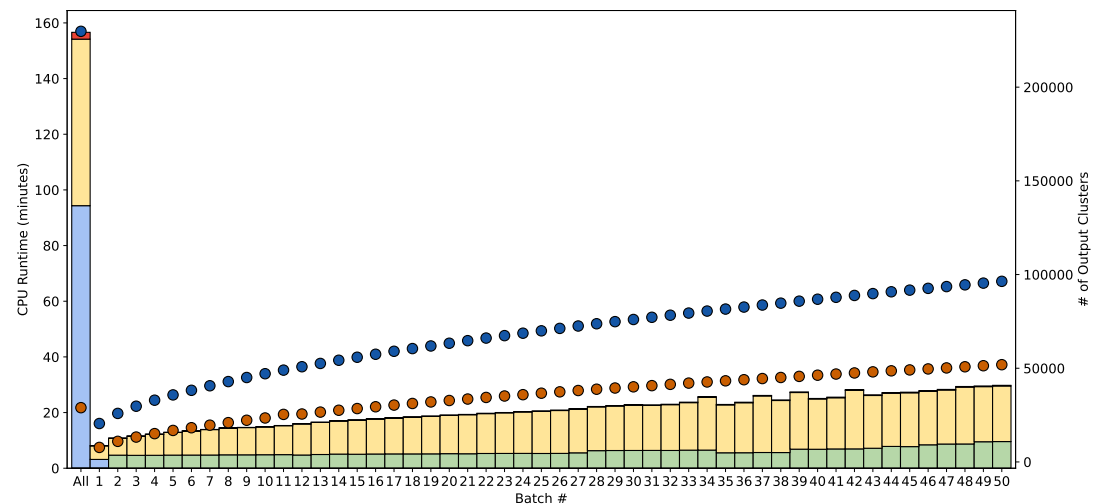



Figure 10. Runtimes of MMseqs2 cascade clustering of sequences in a single batch and 50 increments of MMseqs2 clusterupdate, shown for a 3 million protein sequence data set from the COG database, with and without running Complet+ at each step. While Complet+ takes significantly longer than MMseqs2 clusterupdate, it scales linearly to the number of input clusters.

Batch learning, which is a type of incremental learning, demands that an algorithm should scale linearly with increasing data. We therefore ran several types of experiments to examine the runtime performance of Complet+. Taking the runtimes of all the different modes in SCOPe from Figs. 5 a) and 5 b), we plotted the run times vs. # of clusters input into the algorithm, shown in Fig. 9. We fit a power law to the curves, and SCOPe runtime-inputclusters follows an $x^{1.36}$ while the Swissprot COG curve follows $x^{1.2}$. While it can be interpreted that Complet+ runs polynomial in time as a function of the number of input clusters, there is additional computational complexity. The time is due to the number of input clusters as well as the representative sequence similarity. If more representative sequences are similar than other clustering algorithms, the search will take more time. This is the case with the furthest blue and red point along the x-axis, which are due to the algorithm linclust. linclust yields cluster representatives that are more similar to each other, and therefore, adds another variable in the time complexity. We then conducted another experiment that considered the large dataset of 3 million proteins from the COG database shown in Fig. 10. After splitting it into 50 batches and **only** using the clusterupdate algorithm, the curve (in green) fits a x^1 power law which is exactly linear in time to the number of clusters. Therefore, if the same underlying algorithm is used, Complet+ is linear in time given the number of input clusters; otherwise, the relationship between representative sequences also needs to be taken into account between the algorithms.

Complet+ demonstrates linear time performance on the 50-batch incremental experiment on the full COG database in Fig. 10. Because the current implementation of Complet+ can take 80% to 175% of the original MMseqs2 cluster time, we questioned the scalability of the method. As mentioned, the time that Complet+ takes is a function of how many clusters that the original algorithm (i.e., MMseqs2 clusterupdate) produces, and the distances between cluster representative sequences. Also, cascaded clustering produces many fewer clusters than linclust or single-step clustering, and therefore, running Complet+ halves in time. Complet+ takes less time than the default cascaded algorithm, shown in the single-batch "All" bar of Fig. 10. Complet+ does take longer than clusterupdate on all sequences. However, in incremental learning settings, where batches of sequences are being incrementally added, Complet+ is shown again in in Fig. 10 (in addition to the green curve in Fig. 9) to increase linearly with the amount of input clusters. Therefore, Complet+ is scalable to large datasets in an incremental setting.

5 CONCLUSIONS

Complet+ offers a transparent and easy-to-use solution for merging clusters to improve completeness and reduce the number of potentially redundant, similar clusters. Complet+ demonstrates consistent improvements to completeness with low penalty to homogeneity, all the while possessing linear scalability. Running Complet+ substantially reduces the number of singletons and very small clusters, which is critical for improving the performance of unsupervised methods for biological analysis. Complet+ also produces more biologically representative clusters: achieving a substantial increase in AMI and ARI metrics, which compare predicted clusters to biological classifications. Complet+ is also a versatile tool, improving clustering metrics on clusters generated using a wide range of algorithms, including MMseqs2's single-step, cascade clustering, connected component clustering, and `clusterupdate` modules, as well as `linclust` and CD-HIT. Finally, we show that Complet+ is linearly scalable with respect to the number of clusters being post-processed, by testing it on a COG dataset of over 3 million sequences. The software and source code are available at `https://github.com/EESI/Complet-Plus`.

ACKNOWLEDGMENTS

We would like to thank Drexel University's University Research Computing Facility (URCF) for providing hardware where some of the computations were run.

REFERENCES

Bairoch, A. and Apweiler, R. (1999). The SWISS-PROT protein sequence data bank and its supplement TrEMBL in 1999. *Nucleic Acids Research*, 27(1):49–54.

Bernardes, J. S., Vieira, F. R., Costa, L. M., and Zaverucha, G. (2015). Evaluation and improvements of clustering algorithms for detecting remote homologous protein families. *BMC Bioinformatics*, 16(1):34.

Chandonia, J.-M., Guan, L., Lin, S., Yu, C., Fox, N. K., and Brenner, S. E. (2021). SCOPe: improvements to the structural classification of proteins – extended database to facilitate variant interpretation and machine learning. *Nucleic Acids Research*, 50(D1):D553–D559.

Chen, W., Zhang, C. K., Cheng, Y., Zhang, S., and Zhao, H. (2013). A comparison of methods for clustering 16s rrna sequences into otus. *PLOS ONE*, 8(8):1–10.

Dash, S., Rahman, S. R., Hines, H. M., and Feng, W.-c. (2021). iblast: Incremental blast of new sequences via automated e-value correction. *PLOS ONE*, 16(4):1–16.

Edgar, R. C. (2010). Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, 26(19):2460–2461.

Enright, A. J., Van Dongen, S., and Ouzounis, C. A. (2002). An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):1575–1584.

for Biotechnology Information (NCBI), N. C. Documentation of the blastclust-algorithm. Technical report.

Fox, N. K., Brenner, S. E., and Chandonia, J.-M. (2013). SCOPe: Structural Classification of Proteins— extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Research*, 42(D1):D304–D309.

Galperin, M. Y., Kristensen, D. M., Makarova, K. S., Wolf, Y. I., and Koonin, E. V. (2019). Microbial genome analysis: the cog approach. *Brief Bioinform*, 20(4):1063–1070.

Galperin, M. Y., Wolf, Y. I., Makarova, K. S., Vera Alvarez, R., Landsman, D., and Koonin, E. V. (2021). Cog database update: focus on microbial diversity, model organisms, and widespread pathogens. *Nucleic acids research*, 49(D1):D274—D281.

Halac, M., Sokhansanj, B., Trimble, W. L., Coard, T., Sabin, N. C., Ozdogan, E., Polikar, R., and Rosen, G. L. (2021). Incremental amp; semi-supervised learning for functional analysis of protein sequences. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 01–08.

Hauser, M., Steinegger, M., and Söding, J. (2016). MMseqs software suite for fast and deep clustering and searching of large protein sequence sets. *Bioinformatics*, 32(9):1323–1330.

Hubbard, T. J. P., Murzin, A. G., Brenner, S. E., and Chothia, C. (1997). SCOP: a Structural Classification of Proteins database. *Nucleic Acids Research*, 25(1):236–239.

Karim, M. R., Beyan, O., Zappa, A., Costa, I. G., Rebholz-Schuhmann, D., Cochez, M., and Decker, S. (2020). Deep learning-based clustering approaches for bioinformatics. *Briefings in Bioinformatics*, 22(1):393–415.

Li, W. and Godzik, A. (2006). Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659.

Nguyen, N.-P., Warnow, T., Pop, M., and White, B. (2016). A perspective on 16s rrna operational taxonomic unit clustering using sequence similarity. *npj Biofilms and Microbiomes*, 2(1):16004.

Ozdogan, E., Sabin, N. C., Gracie, T., Portley, S., Halac, M., Coard, T., Trimble, W., Sokhansanj, B., Rosen, G., and Polikar, R. (2021). Incremental and semi-supervised learning of 16s-rrna genes for taxonomic classification. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7.

Paccanaro, A., Casbon, J. A., and Saqi, M. A. S. (2006). Spectral clustering of protein sequences. *Nucleic Acids Research*, 34(5):1571–1580.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Romano, S., Vinh, N. X., Bailey, J., and Verspoor, K. (2015). Adjusting for chance clustering comparison measures.

Rosenberg, A. and Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic. Association for Computational Linguistics.

Schloss, P. D. and McMahon, K. (2021). Amplicon sequence variants artificially split bacterial genomes into separate clusters. *mSphere*, 6(4):e00191–21.

Steinegger, M. and Söding, J. (2017). Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnology*, 35(11):1026–1028.

Steinegger, M. and Söding, J. (2018). Clustering huge protein sequence sets in linear time. *Nature Communications*, 9(1):2542.

Vanni, C., Schechter, M. S., Acinas, S. G., Barberán, A., Buttigieg, P. L., Casamayor, E. O., Delmont, T. O., Duarte, C. M., Eren, A. M., Finn, R. D., Kottmann, R., Mitchell, A., Sánchez, P., Siren, K., Steinegger, M., Gloeckner, F. O., and Fernàndez-Guerra, A. (2022). Unifying the known and unknown microbial coding sequence space. *eLife*, 11:e67667.

Vinh, N. X., Epps, J., and Bailey, J. (2009). Information theoretic measures for clusterings comparison: Is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1073–1080, New York, NY, USA. Association for Computing Machinery.

Wong, S. and Ragan, M. A. (2008). Machos: Markov clusters of homologous subsequences. *Bioinformatics*, 24(13):i77–85.

Youngblut, N. D. and Ley, R. E. (2021). Struo2: efficient metagenome profiling database construction for ever-expanding microbial genome datasets. *PeerJ*, 9:e12198.

Zhao, Z., Cristian, A., and Rosen, G. (2020). Keeping up with the genomes: efficient learning of our increasing knowledge of the tree of life. *BMC Bioinformatics*, 21(1):412.