Efficient algorithms for accurate hierarchical clustering of huge datasets: tackling the entire protein space

Yaniv Loewenstein\textsuperscript{1,*}, Elon Portugaly\textsuperscript{1}, Menachem Fromer\textsuperscript{1} and Michal Linial\textsuperscript{2,*}

\textsuperscript{1}School of Computer Science and Engineering and \textsuperscript{2}Department of Biological Chemistry, Institute of Life Sciences, The Hebrew University of Jerusalem, Israel

\section*{ABSTRACT}

\textbf{Motivation:} UPGMA (average linking) is probably the most popular algorithm for hierarchical data clustering, especially in computational biology. However, UPGMA requires the entire dissimilarity matrix in memory. Due to this prohibitive requirement, UPGMA is not scalable to very large datasets.

\textbf{Application:} We present a novel class of memory-constrained UPGMA (MC-UPGMA) algorithms. Given any practical memory size constraint, this framework guarantees the correct clustering solution without explicitly requiring all dissimilarities in memory. The algorithms are general and are applicable to any dataset. We present a data-dependent characterization of hardness and clustering efficiency. The presented concepts are applicable to any agglomerative clustering formulation.

\textbf{Results:} We apply our algorithm to the entire collection of protein sequences, to automatically build a comprehensive evolutionary-driven hierarchy of proteins from sequence alone. The newly created tree captures protein families better than state-of-the-art large-scale methods such as CluSTr, ProtoNet\textsuperscript{4} or single-linkage clustering. We demonstrate that leveraging the entire mass embodied in all sequence similarities allows to significantly improve on current protein family clusterings which are unable to directly tackle the sheer mass of this data. Furthermore, we argue that non-metric constraints are an inherent complexity of the sequence space and should not be overlooked. The robustness of UPGMA allows significant improvement, especially for multidomain proteins, and for large or divergent families.

\textbf{Availability:} A comprehensive tree built from all UniProt sequence similarities, together with navigation and classification tools will be made available as part of the ProtoNet service. A C++ implementation of the algorithm is available on request.

\textbf{Contact:} lonshy@cs.huji.ac.il

Hierarchical connections are especially evident in the biological domain. Gene Ontology (GO; Ashburner \textit{et al.}, 2000) classifies genes into hierarchies of biological processes and molecular functions. The SCOP, CATH and DALI databases classify protein structures into a hierarchy based on structural similarities (Murzin \textit{et al.}, 1995). The ENZYME commission (EC) nomenclature classifies enzymes into a hierarchy based on biochemical classes. Classical taxonomy classifies organisms into an evolutionary tree structure. Notably, the evolutionary tree process driving sequence divergence, underlies the hierarchical classification of protein families, and make it an especially appealing playground for hierarchical clustering methods. UPGMA (Unweighted Pair Group Method using arithmetic Averages) is arguably the most popular hierarchical clustering algorithm in use to date, especially for gene expression (D'haeseleer, 2005) and for protein sequence clustering (Liu and Rost, 2003).

One of the daunting problems in the field of computational biology is the development of automatic methods for structure and function prediction from protein sequence. This challenge is emphasized by the glut of protein sequences deposited in public databases (Suzek \textit{et al.}, 2007).

ProtoNet (Kaplan \textit{et al.}, 2004, 2005) uses UPGMA to build a hierarchy of protein sequences from sequence similarities. It was shown that this automated procedure is especially useful for prediction of function (Sasson \textit{et al.}, 2006), remote homology (Shachar and Linial, 2004) and structure (Kifer \textit{et al.}, 2005). Since the clustering is unsupervised, it is independent of available knowledge, and can thus automatically unveil clusters of novel biological significance. Other large-scale unsupervised (i.e. using only sequence) methods are Systers (Krause \textit{et al.}, 2005) and CluSTr (Petryszak \textit{et al.}, 2005), which utilize single-linkage clustering to cope with the data size.

\section*{1 INTRODUCTION}

\subsection*{1.1 Background}

Clustering is a fundamental task in automatic processing of large datasets, in a broad spectrum of applications. It is used to unravel latent natural groupings of data items. Hierarchical clustering methods aim to furthermore categorize data items into a hierarchical set of clusters organized in a tree structure. For instance hierarchical clustering is used for automatic recognition and classification of patterns in digital images, stock prediction, text mining and in computer science theory.

Hierarchical clustering algorithms construct a hierarchy of input data items. Agglomerative clustering methods create a hierarchy bottom-up, by choosing a pair of clusters to merge at each step. The result is a rooted binary tree. $N$ leaves correspond to input data items (singleton clusters), and $N-1$ inner nodes (clusters) correspond to groupings in coarser granularities at higher tree levels. Merge scores correspond to dendrogram heights. The hierarchy is often used to infer knowledge from cluster statistics, as well as relatedness at varying granularities.

Agglomerative clustering methods usually take an input of pairwise similarities among data items, from which cluster similarities are then inferred. Different formulations have been used to define pairwise similarities across clusters. Single-linkage
Table 1. Size of clustered UniRef90 data, extrapolated future sizes and the appropriate memory requirements

|                         | Memory (GB) | | Edges (No. of edges) |
|-------------------------|------------|---|---------------------|
| **Current – UniRef90 rel. 8.5, June 2006** | 55         | 2.5 \times 10^{12} |
| Rare BLAST similarities (directed multi-graph) | 50         | 2.5 \times 10^{12} |
| Actual sparse similarities (symmetric, undirected) | 30         | 1.5 \times 10^{12} |
| Full non-sparse possible similarities | 12,981     | 6.0 \times 10^{12} |
| **Extrapolated Future – UniRef90 rel. 13.1, March 2008** | 1.80M      | 1.80M |
| Actual sparse similarities (symmetric, undirected) | 122        | 6.1 \times 10^{12} |
| Full non-sparse possible similarities | 52,262     | 6.5 \times 10^{12} |

Expected memory requirements are based on a conservative estimate of 20 bytes per edge for sparse data, and 8 bytes for full matrices. Extrapolation is based on the actual growth in the number of sequences in UniRef90, when a fixed degree of sparsity is assumed. A strong 32-bit workstation has up to 4 GB of physical memory, which in practice can hold about 40–200M edges.

1.3 Goal

We aim to provide a practical UPGMA algorithm which is not limited by memory requirements but guarantees the correct clustering solution. Non-metric considerations are of special interest, as shown for the case of protein sequence clustering. We thus set out to develop a general clustering framework for any type or size of data (not necessarily metric), while maintaining the relative simplicity of hierarchical clustering.

Y. Loewenstein et al.
2.2 Sparse UPGMA

$E$ is considered sparse if not all possible pairs exist in the input graph $(\exists i,j \in V: e_{ij} \notin E)$. For instance, the sets $(E)$ of all BLAST sequence similarities, or all protein–protein interactions, are sparse.

In this section, we formulate UPGMA for sparse inputs, and give a suitable algorithm—Sparse-UPGMA (Fig. 2). We take advantage, in time and memory complexity, of the data’s sparsity, to allow clustering of large sets. It is correct and efficient for non-sparse inputs as well. Based on Sparse-UPGMA, in the next sections we present a novel class of memory-constrained UPGMA (MC-UPGMA) algorithms which do not require all edges in memory at once.

### 2.2.1 Missing edges

In UPGMA the arithmetic mean is used to measure distance across clusters.

$$d_{ij} = \frac{1}{|C_i||C_j|} \sum_{p \in C_i \cap C_j} d_{pq}$$

Equation (1) (or any other average-linkage formulation) is not well defined for sparse inputs (i.e. when $e_{pq} \notin E$). To expand $d$’s domain to all vertex pairs (i.e. now $d: \forall V \times \forall \rightarrow \mathbb{R}^+$), we override the previous notation to introduce a missing value completion rule.

$$\psi = \begin{cases} \max(E) & d_{ij} \notin E \\ \frac{1}{|C_i|} \sum_{p \in C_i \cap C_j} d_{pq} & \text{otherwise} \end{cases}$$

Where input $d_{ij}$’s are used when available and $\psi$ is used otherwise. Hence, $\psi$ is a detection threshold used to account for missing edges in the average distance across two partially connected clusters. It does not imply that all clusters are connected; rather, it is used only in the average distance across two partially connected clusters. It decreases the memory requirement considerably. Furthermore, new thick edges can efficiently be computed recursively (in $O(1)$). This calculation, denoted as $\forall \psi \mathbf{F}^{(\bullet, \bullet)}$ hereby, requires $\psi$, cluster sizes $|C_i|$, and two precalculated edges. For Sparse-UPGMA we have:

$$d_{ij} = \begin{cases} \max(E) & e_{ij} \notin E \\ \frac{1}{|C_i|} \sum_{p \in C_i \cap C_j} d_{pq} & \text{otherwise} \end{cases}$$

A similar $O(1)$ expression is easily derived and plugged into our algorithms for other (non-UPGMA) formulations (e.g. for the geometric mean).

### 2.2.2 Leveraging sparsity

For suitable inputs, a sparse representation for $E$ decreases the memory requirement considerably. Furthermore, new thick edges can efficiently be computed recursively (in $O(1)$). This calculation, denoted as $\forall \psi \mathbf{F}^{(\bullet, \bullet)}$ hereby, requires $\psi$, cluster sizes $|C_i|$, and two precalculated edges. For Sparse-UPGMA we have:

$$d_{ij} = \begin{cases} \max(E) & e_{ij} \notin E \\ \frac{1}{|C_i|} \sum_{p \in C_i \cap C_j} d_{pq} & \text{otherwise} \end{cases}$$

A similar $O(1)$ expression is easily derived and plugged into our algorithms for other (non-UPGMA) formulations (e.g. for the geometric mean).

### 2.2.3 Complexity and scalability

Using a suitable binary heap data structure for maintaining $E$ sorted, Sparse-UPGMA requires $O(t\log|V|)$-time and $O(t\log|V|)$-memory algorithm. These improvements allow for clustering of considerably large sparse datasets. For instance, it can cluster the sparse 75K×75K set of mouse cDNA measurements, that could not be hierarchically clustered in the supplementary data of Frey and Dueck (2007). We were also able to cluster the ProteinNet4 data (114K) requiring time (3 min on a 2.80GHz machine) that was negligible as compared to the similarity computation preprocessing time. However, this algorithm could not cope with huge datasets, where an $O(|E|)$ memory requirement is intolerable (e.g. Table 1). Due to poor locality of reference, this algorithm is rendered impractical when the virtual memory demand exceeds the physically available memory.

### 2.3 Exact (correct) clustering

We define a (UPGMA) clustering solution as exact (or correct), if the order of merges is correct (up to equidistant merges), i.e. it always yields the same solution as UPGMA (or Sparse-UPGMA for sparse inputs), regardless of computational limitations such as memory requirements.

### 2.4 Multi-Round MC-UPGMA

#### 2.4.1 Outline

We introduce the concept of MC-UPGMA. The proposed solution, breaks the clustering process into multiple rounds. Two computation units carry out each round (Fig. 3). A memory constrained clusterer, holding only a subset of $E$ that fits in memory, outputs successive parts of the overall hierarchy. The second unit, the merger (Fig. 3), is a modular unit, external to the clustering, but memory constrained as well. It processes the partial clustering and the set of current edges, to produce valid edges. Edges grow thicker as clusters grow larger. The current set of valid thick edges is input for the clusterer and is used to resume clustering in the successive round.

#### 2.4.2 Memory-constrained clustering

Round $t$ of clustering starts from an input (sparse) set of edges, $E_t$, between valid (unmerged) clusters. Initially, $E_1 = \emptyset$. Clustering is governed by a fixed memory budget parameter $M$, denoting the maximal number of maintained edges (based on memory size). At initialization, the clusterer loads only the best (minimal) $M$ edges denoted by $E_1$. Higher $M$s may require fewer rounds (due to additional progress per round). We also define $\lambda_{\max}$ maximum of (at most $M$) loaded edges at round $t$
2.4.3 Incorrect naive clustering At first, it is compelling to naively cluster all loaded edges. We give a simple counter example which will create a wrong tree when this is done. The intention is to motivate construction of a proper stop criterion thereafter.

Assume $\lambda = 10$ and $\phi = 100$. Let $C_1$ and $C_2$ be some clusters. Suppose $\lambda$ is merged with some $C_2 \neq C_1$ at height $d_{kl} = 7$. Then, $C_1$ may not have been the closest neighbor, e.g. $d_{kl} = 10$ (since $d_{kl} = 10$ but $d_{kl} = 6 < d_{kl} = 7$). On the other hand, if $C_1$ and $C_2$ are indeed merged, then set $d_{kl} = \phi = 100$ and thus merging is still incorrect ($d_{kl} = 7 < d_{kl} = 100$).

Hence, a correct clusterer should be mindful of unseen edges ($\geq \lambda$), affecting clustering before $\lambda$. Such examples are rather prevalent in non-metric datasets. Figure 4 portrays the relevance of this example for the case of clustering sequence similarities.

2.4.4 Uncertain edge intervals To prevent false clustering of a non-minimal edge, we maintain suitable bounds per edge. The value $d_{lk}$ (since $e_{ij}$) is the exact value of $d_{ij}$. Suppose $\lambda$ halts when it is impossible to identify the minimal edge in the uncertain edge values (due to partial edge data at hand). Clustering non-metric datasets. Figure 1 portrays the relevance of this example.

2.4.5 Clusterer algorithm The clustering algorithm (Fig. 4) is a modified version of the code for Sparse-UPGMA (Fig. 2).

2.4.6 Stop criterion A potential merge ($e_{ij}$) in Fig. 4 may not be provably minimal if (A) a smaller edge may exist outside the clusterer memory scope $E'$ or (B) the minimal edge may be any smaller edge. However, the latter case (B) manifests itself as edge interval clashes. To assure that $d_{ij}$ is provably minimal, we consider its interval.

Clustering proceeds while a distinctly minimal edge is at hand—a non-loaded edge whose upper bound $u_{ij}$ is below the lower bound of any other edge in $E'$ (i.e. $e_{ij}$ $\in$ $E' \setminus \lambda$ in the case where $(i, j) \neq (r, s)$). By maintaining the criterion $\lambda \geq \lambda_{ij}$ in the case where $(i, j) \neq (r, s)$ we assert that the output merge values $d_{ij}$ are exact. If only correct merge order, but not exact merge values is required, this criterion can be relaxed to apply only when $(i, j) \neq (r, s)$. In order to construct a full dendrogram (with heights) for the studied proteins, we have used the harsher criterion throughout our analysis.

2.4.7 Correctness and progress The clusterer always progresses, since after initialization all edges are exact and $\leq \lambda$. Furthermore, progress is optimal for this setting, since we have shown a counter example that falsifies the algorithm if it does not halt. Therefore, the number of clusters is reduced in each round, and $|E'|$ is accordingly reduced at a quadratic rate. Once all edges fit in memory $(|E'| \leq \lambda)$, all edge intervals become exact, and Multi-Round MC-UPGMA reduces to Sparse-UPGMA. The clusterer maintains the loop invariant that $e_{ij}$ is minimal over all edges, if the stop criterion has not been met. Combined with the fact that the clusterer always makes some progress, we conclude that the correct tree is output.

2.4.8 Progress guarantee—metric setting If the data obeys the triangle inequality, then further clustering progress can be made in each round. The clustering is guaranteed to progress well in this metric setting, so that the tree is complete within only very few rounds. We provide a short claim that shows how good progress can be provably guaranteed for the first iterations. We aim to show how constraints implied from metric considerations render the clustering easier. With some technical rigor, it is possible to generalize this claim.

**METRIC PROGRESS LEMMA.** If input edges satisfy the triangle inequality, then Multi-Round MC-UPGMA clusters all edges $\leq \lambda$. Proof. Exact minimal edges do not halt clustering. We will show that inexact edges appear only after $\lambda$. Let $e_{ij} \in E'$ be an inexact edge appearing along the clustering process, and assume w.l.o.g. }
that $C_3$ was created before $C_2$ is merged. Let $C_2$ and $C_1$ be the clusters merged to create $C_3$, i.e. $C_3 = C_2 \cup C_1$. Either $d_{ij} \leq \lambda$ or $d_{ij} \geq \lambda$, otherwise $d_{ij} > \lambda \Leftrightarrow E_{ij} \notin E$. Assume w.l.o.g. that only $d_{ij} \leq \lambda$ (i.e. $d_{ij} \leq \lambda$), since otherwise $ekl$ would have been inexact. From merge order, we have $d_{ij} \leq d_{ij} \leq \lambda$. Now, note that $d_{ij} \geq \lambda$, otherwise $d_{ij} \leq d_{ij} < \lambda$ implying $d_{ij} \leq d_{ij} + \lambda < \frac{\lambda}{2} = \lambda$ (due to the triangle inequality), contradicting our assumption. Plugging in Equation (2), we have $ekl = \frac{\lambda}{2} - (\lambda + \lambda) = \frac{\lambda}{2}$.

The progress guarantee is given on the height axis of the forming tree. This translates to very good progress when the data is exponentially distributed (i.e. edges are orders of magnitude different). For instance, this is the case for BLAST similarity $\gamma$-values. Note that the triangle inequality assumption is never used explicitly by the algorithm, but only to characterize its worst-case progress—the algorithm is correct regardless. Furthermore, if the triangle inequality is used explicitly in Multi-Round MC-UPGMA, the rather crude global $\gamma$ bound can be replaced with an edge dependent bound (using the triangle), to reduce clashes and allow further clustering per round. From our experiments (data not shown), the metric assumption can sustain some (bounded) noise, and still yield good progress, i.e. if $ekl \leq (1+\varepsilon)(ekl + ejl)$ is satisfied for some fixed $\varepsilon > 0$.

2.5 External edge merging

Due to space limitations, we discuss external edge merging very briefly. We then quickly turn to a clustering algorithm which does not need it in the next section.

The merger unit collates edges of newly merged clusters, into thicker edges between the respective parents in the forming tree (as in \textsc{LoadEdges}, Fig. 5). Merging requires the previous set of edges $E^{t-1}$ and the forming tree. A naive algorithm will form $E^t$ in memory, thus invalidating the memory constraint. If edges were read in the correct order, however, it is possible to collate one thick edge at a time in memory. This is achieved by appropriate on-disk sorting of only modified edges in $E^t \setminus E^{t-1}$. Sorting may be prohibitedly slow when more than a few rounds are required however. The two ideas can be combined to do some limited merging in memory. Notably, merging can be distributed to any number of parallel merging processes. It requires that all edges composing a single thick edge be delivered to the same merging process. This is achieved by mapping edges to processes based on a hash function that is mindful of the current cluster indices to which edges belong.

2.6 Single-Round MC-UPGMA

2.6.1 Motivation

Up until now, MC-UPGMA built the clustering tree round by round. Although this yielded a practical solution, most of the computation time is spent on preprocessing for the next round of clustering. We address this issue by devising a MC-UPGMA scheme that clusters the entire dataset in a single round.

2.6.2 Approach

Here we aim to combine ideas from the algorithms for external computation of valid edges, together with the idea of careful clustering with $\lambda$-missing edges and uncertainty intervals. Clearly, when Multi-Round MC-UPGMA is used, it is not using its entire memory budget $M$, since each merge reduces the number of edges in memory. The newly formulated hybrid algorithm presented in this section, will use the freed-up memory to load fresh edges (up to $M$ edges in total) from disk. This reloading enables the clustering to proceed. We assume that input edges are sorted on-disk, and are loaded in order of ascending $d_{ij}$ values.

An immediate difficulty for reading edges after doing some clustering is imposed by reading "old" invalid edges—those involving clusters which have already been merged. The Single-Round MC-UPGMA algorithm addresses this difficulty. To accommodate edge reading after some clustering has been done, we introduce a new edge representation, that is invariant to the ongoing merging process.

Rereading more edges allows further clustering, where the previous algorithm had to halt. First, once more edges are read, the value of $\lambda$ increases dynamically and is no longer fixed. Hence, $\lambda$-dependent recalculating of $\gamma$’s reduces interval clashes, and clustering may proceed. Furthermore, uncertain edges resulting from a missing edge can now be updated with certainty to the exact $d_{ij}$ value, by reading previously missing information, from disk.
We think of

The triangle inequality guarantees that missing edges are loaded.

The number of sequences affects the volume and especially time for the intensive reloading edges from disk, which become certain (\(\lambda\)), previously replaced with a fixed \(\lambda\), now dynamically updated with the current \(\lambda\)-value:

\[
\tilde{c}_{ij} = \text{edge sum}(d_{ij}) + \lambda \left(|C_i| - |C_j| - \text{count}(d_{ij})\right) \\
\tilde{u}_{ij} = |C_j| - |C_i| - \text{count}(d_{ij})
\]

We think of \(|C_i| - |C_j| - \text{count}(d_{ij})\) as a dynamic uncertainty weight. Now, when a missing component of an uncertain edge is read while reloading edges from disk, \text{count}(d_{ij}) is incremented. Consequently the respective uncertainty weight diminishes, and the interval tightens up around \(d_{ij}\). Fully linked edges (i.e. \text{count}(d_{ij}) = |C_i| = |C_j|) become certain ('\(d_{ij}\)'). For uncertain edges, missing components of the sum are replaced by the current tightest bound.

Since edges are read in ascending order, \(\lambda\) grows as more edges are loaded. Because uncertain intervals are computed dynamically, lower bounds become tighter as \(\lambda\) grows. Consequently, intervals tighten-up, and edge interval clashes are reduced as more and more edges are loaded. Hence, reloading allows further clustering.

2.6.4 Progress If the data is metric, the per-round-progress proof shown by the triangle inequality is extended to show that clustering in one round is possible. The reasoning is similar. The triangle inequality guarantees that missing edges are loaded promptly to allow clustering progress. For hard non-metric input, it is theoretically possible that clustering still can not proceed after reloading, while the entire memory budget \(M\) is in use. To assure progress, we introduce a look-ahead procedure after which all edges become exact and clustering can effectively resume (Fig. 5).

2.6.5 Complexity Single-Round MC-UPGMA now requires \(O(N)\) (typically \(N < M\)) memory for holding the forming tree. The path compression heuristic allows for efficient cluster look-up.

3 METHODS

3.1 Clustered sequence data

We undertake the comprehensive set of all proteins in the UniProt (release 8.1) (Suzek et al., 2007), composed of Swiss-Prot (rel. 50.1) and TrEMBL (rel. 33.1) proteins. UniRef90 (rel. 8.5) non-redundant (<90% identity) sequences were used to represent the protein space. Non-UniRef90 sequences are redundant since they (1) show the same similarity patterns with the rest of the data, thus adding no clustering relevant information and (2) can be regarded as functionally equivalent (Lou and Rost, 2003). The number of sequences affects the volume and especially time for the intensive computation of all-against-all similarities, rather than the clustering.

3.2 BLAST sequence similarities

All sequences in the UniRef90 set were compared using the blastp program of the BLAST (Altschul et al., 1997) 2.2.16 suite, using a \(K=100\) threshold, low-complexity filtering and default parameters (BLOSUM62, -11,-1). BLAST runs were executed in parallel using a MOSIX grid, as part of the new ProtoNet standard build process.

Sequences were compared using a reciprocal-BLAST-like setting where each sequence is used both as query and database entry. The result is a directed multigraph. It is then reduced to an undirected graph (symmetric dissimilarities) by keeping only \(\psi_{ij} = \min(\psi_{ij}, \psi_{ji})\) for \(i<j\), i.e. half of the non-triangular all-against-all sparse dissimilarity matrix. The data sizes before and after this processing are shown in Table 1. Relying on the high capacity of our novel algorithm for very large edge sets, we allow a very permissive threshold (\(E=100\)). This allows for more edges to guide the clustering process, especially for the cases of barely detectable similarities. We rely on UPGMA's robustness to filter out noise manifested as spurious non-significant edges, or amplify weak but consistent similarities by averaging over large clusters. For the case of single-linkage clustering (which is part of our comparative evaluation), including low-significance edges does not interfere with performance either, since they are used only after more significant edges are utilized.

3.3 Protein family keywods

To evaluate the quality of a clustering solution, we measure the correspondence of a tree to external expert classifications of protein families. Here, we use the InterPro (rel. 12.1) (Mulder et al., 2007) classification of protein families as a mapping of keywords to protein sequences. InterPro is a consortium of protein families derived from member databases of protein sequence signatures such as Pfam (Finn et al., 2006). InterPro further categorizes keywords into (1) InterPro domains which appear in the context of at least two different non-overlapping protein signatures, and are thus considered a modular protein fragment (positional) and (2) Protein families which refer to a group of proteins in a match set (whole proteins, rather than a sub-region). For this study we have used keywords which incident on at least 2 (10) UniRef90 proteins, a total of 3752 (3528) are \(\geq 10\) InterPro domains, and 8965 (7047) families.

3.4 Performance metrics

A single protein might be associated with multiple keywords; e.g. as in the case of multi-hetero-domains. In the context of a particular keyword \(k\) (e.g. InterPro accession IPR001267—thymidine kinase) and the cluster \(C_k\), a protein in \(C_k\) is regarded as a true positive (TP) if it has the particular keyword, and as a false positive (FP) if it has a keyword, but it does not have \(k\). Classified proteins outside the cluster, having or not having the particular keywords, are regarded as false and true negatives (FN and TN, respectively). Proteins having no keywords participate in the clustering, but do not affect the evaluation.

A cluster is assigned three quality measures. Specificity (\(=\frac{TP}{TP + FP}\)) and sensitivity (\(=\frac{TP}{TP + FN}\)) measure the accuracy of a cluster with respect to cluster members or the reference keyword, respectively. Tree leaves (root), contain a single (all) protein(s), and therefore trivially have full specificity (sensitivity). However, neither convey interesting groupings. A clustering captures a protein family keyword \(k\) well, if it contains a cluster having both high specificity and sensitivity for \(k\). This is captured by assigning each cluster-keyword pair a correspondence-score which accounts for both specificity and sensitivity.

\[
\mathcal{J}(C, k) = \frac{|C \cap \mathcal{A}(k)|}{|C| + |\mathcal{A}(k)|} = \frac{TP}{TP + FP + FN}
\]

This set-theoretic inspired score (Jaccard score) is a standard clustering performance metric (Kaplan et al., 2005; Krause et al., 2005). The value of \(\mathcal{J}\) ranges from 0 for no correspondence (intersection), to 1 for full agreement when specificity = sensitivity = 1. Since \(\mathcal{J} \leq \text{specificity} \leq \text{sensitivity}\), it is a harsh performance metric.
3.5 Best cluster
To evaluate the tree with respect to a particular keyword \( k \), we select the corresponding best cluster. For keyword \( k \), this policy is reflected by taking

\[
J(k) = \max_{C \subseteq \mathcal{C}, |C| = 1} J(C, k)
\]

(6)

Singleton clusters are omitted, since they do not convey information about classification or clustering quality. By balancing specificity and sensitivity, this score (Equations (5) and (6)) effectively selects against clusters near the root or leaf. This scheme calibrates the groupings (clusters) for the required evolutionary granularity, and selects against intermediate clusters—partial groupings which are artifacts of clustering into a binary tree. It selects biologically meaningful clusters, e.g. with respect to protein family size. To evaluate the tree with respect to a set of keywords \( K \) (e.g. InterPro domains), we select the best cluster per keyword (Equation (6)), and average across \( K \) using either equal weights (family-centric) or denote by \( J^* \) a family-size-weighted average (protein-centric).

3.6 Comparison with other methods
To evaluate the contribution of the newly formulated UPGMA tree of all protein sequences, we compare it with trees resulting from other methodologies. We aim to assess the contribution of our work on very large datasets, rather than to assess various clustering methods. We test the clustering performance for different sequence databases, reflecting different difficulty levels and data sizes. We compare MC-UPGMA with three other methodologies which could be applied to this size of data.

3.6.1 CluSTr slim (Petryszak et al., 2005)—contains the CluSTr pruned tree, downloaded from the EBI ftp. CluSTr uses an in-house single-linkage clustering algorithm, applied to pairwise similarities derived from Smith-Waterman alignments of UniProt (rel. 12.5) and genomic sequences. CluSTr applies a significantly more conservative similarity threshold of \( E = 1 \times 10^{-40} \) than \( E = 100 \) CluSTr Slim is a pruned tree of clusters \( \leq 1000 \) with \( \varepsilon \)-clusters (clusters with \( \varepsilon \)-linkages) which maintains CluSTr’s predictive power (Petryszak et al., 2005). Removal of proteins in isolated clusters (not in the hierarchy) or the root cluster did not alter results significantly.

3.6.2 Single-linkage clustering controls for the effects of different alignment algorithm (BLAST versus Smith-Waterman) and similarity thresholds compared to CluSTr. This method applied an in-house single-linkage implementation to our BLAST similarity data.

3.6.3 ProtoNet4 protocol uses sparse UPGMA clustering to build a tree skeleton from a reduced-size set of high-quality Swiss-Prot proteins, which are regarded as protein family representatives. Sequences from the much larger UniRef90 set (or TrEMBL), are then appended to the existing skeleton independently, based on similarities to sequences in the reference skeleton. It can thus be regarded as a representative-based heuristic for clustering of very large sets. Accordingly, this method is unable to capture protein families which are not represented well in the Swiss-Prot skeleton. This method does not use similarities within the larger set, but only the manageable smaller clustered set.

4 RESULTS AND DISCUSSION
Sneath (1957) offered the application of computers to taxonomy. 10 years later and some 40 years ago, Fitch and Margoliash (1967) provided what was probably the first automated evolutionary tree for the largest available family at the time—twenty cytochrome-c sequences. Here we tackle the challenge of accurate clustering of nearly 2 million non-redundant sequences, to build a comprehensive tree which aims to capture the evolutionary processes underlying protein families.
Table 2. Clustering performance evaluation based on InterPro keywords

| Dataset | InterPro Families | InterPro Domains |
|---------|------------------|------------------|
|                | $\mathcal{J}$ | Spec. Sens. | $\mathcal{J}$ | Spec. Sens. |
| UniRef90       |                |              |                |              |
| MC-UPGMA (current) | 0.900 | 0.965 | 0.926 | 0.735 | 0.895 | 0.798 |
| CluSTr Slim    | 0.280 | 0.934 | 0.292 | 0.239 | 0.881 | 0.253 |
| Single-linkage | 0.808 | 0.952 | 0.842 | 0.566 | 0.878 | 0.619 |
| ProtoNet4 protocol | 0.795 | 0.941 | 0.832 | 0.669 | 0.901 | 0.720 |
| UniRef50       |                |              |                |              |
| MC-UPGMA (current) | 0.881 | 0.959 | 0.911 | 0.717 | 0.887 | 0.782 |
| Single-linkage | 0.794 | 0.947 | 0.830 | 0.557 | 0.877 | 0.608 |
| Swiss-Prot     |                |              |                |              |
| MC-UPGMA (current) | 0.935 | 0.980 | 0.952 | 0.809 | 0.948 | 0.842 |
| Single-linkage | 0.911 | 0.968 | 0.938 | 0.747 | 0.941 | 0.783 |
| CluSTr Slim    | 0.470 | 0.955 | 0.489 | 0.375 | 0.889 | 0.406 |

50%). Swiss-Prot (220K) reflects a moderately sized high-quality set, with some trivial redundancies. All methods, excluding CluSTr Slim and ProtoNet4, were benchmarked by clustering the respective set alone. CluSTr Slim performance on UniProt (redundant, contains trivial cases) is given for reference. CluSTr Slim performance is based on an evaluation based only on UniRef90 representatives, but clustering is based on all UniProtKB proteins.

Table 3. Average UniRef90 performance, unweighted ($\mathcal{J}$) and weighted by non-redundant protein family size ($\mathcal{J}^w$)

|                | InterPro Families | InterPro Domains |
|----------------|------------------|------------------|
|                | $\mathcal{J}$ | $\mathcal{J}^w$ | $\mathcal{J}'$ |
| UniRef90       |                |              |                |
| MC-UPGMA (current) | 0.900 | 0.856 | 0.735 |
| Single-linkage | 0.808 | 0.599 | 0.566 |
| CluSTr Slim    | 0.470 | 0.489 | 0.375 |
| ProtoNet4 protocol | 0.795 | 0.782 | 0.669 |

Table 4. Clustering progress for the hard UniRef90 data

|                 | No. of merges | Last merge | $\mathcal{J}'$ | $\mathcal{J}^w$ | $\mathcal{J}'$|\$/\mathcal{J}^w$ (%) |
|-----------------|--------------|-----------|---------------|---------------|---------------|------------------|
| 1               | 679915       | 6e−09     | 0.55e±0.08    | 36.77         |
| 2               | 450486       | 1e−24     | 1.110401     | 3.15e±0.08    | 20.77         |
| 3               | 258122       | 8e−05     | 1.368523     | 2.24e±0.08    | 14.78         |
| 4               | 20774        | 0.004     | 1.395297     | 2.14e±0.08    | 14.06         |
| 5               | 39712        | 0.444     | 1.435509     | 1.80e±0.08    | 12.47         |
| 6               | 11534        | 1.052     | 1.446543     | 1.82e±0.08    | 11.98         |
| 7               | 5225         | 1.474     | 1.451768     | 1.76e±0.08    | 11.75         |
| 8               | 5948         | 2.062     | 1.435716     | 1.75e±0.08    | 11.50         |
| 9               | 11011        | 3.533     | 1.468727     | 1.66e±0.08    | 11.03         |
| 10              | 4286         | 4.277     | 1.473013     | 1.65e±0.08    | 10.88         |

Fig. 6. Sensitivity of the best cluster for single-linkage ($x$-axis) versus MC-UPGMA ($y$-axis) for large protein families—InterPro domains and families with at least 100 UniRef90 representatives. Diagonal line represents identity ($y=x$). Points above diagonal correspond to higher sensitivity for MC-UPGMA and vice versa. Average specificities are comparable across this set (MC-UPGMA $= 0.90 \pm 0.15$, single-linkage $= 0.88 \pm 0.18$).

4.4 Clustering detects poorly connected families

Out of 10,808 InterPro keywords which cover more than 10 proteins in UniRef90, 8218 are captured very well ($\mathcal{J} \geq 0.70$). Of the 7992 best clusters for these families, only 2467 are fully linked (i.e. not sparse), 2793 are $<50\%$ linked, and 792 clusters are highly divergent and are $<10\%$ linked. Yet, all are picked up by our method with high accuracy. This demonstrates the capacity of the method to pick up even highly divergent protein families. The latter are dominated by homologous pairs which are not detectable by even a very permissive BLAST threshold, yet MC-UPGMA is able to pick them up, leveraging transitive similarities in large clusters.

4.5 Analysis of multi-round MC-UPGMA run

The Multi-Round MC-UPGMA algorithm applied to the UniRef90 set required 200 clustering rounds overall (Table 4). Using a single 4 GB memory 4-CPU workstation, we are able to parallelize external merging, and tolerate multiple clustering rounds to cluster the whole set within about 1–2 days. This is orders of magnitude less than the CPU-time required for preprocessing—computation of all BLAST sequence similarities. Additional speedup is possible using grid-computing. In the first three rounds 76% of the clustering (reducing the edge data by 85%) was done. Although the triangle inequality does not hold even for these first rounds, the algorithm is able to sustain some metric distortion and still progress well. The clustering becomes computationally challenging only in the absence of edges, in otherwise connected clusters. The clustering progress per round is significantly slowed down when the triangle inequality is strongly violated due to non-existent BLAST edges (first encountered after 1.37 M merges out of total 1.8 M). Initial tests indicate that Single-Round MC-UPGMA is able to considerably speed up the process (not shown).

4.6 Protein sequence space inherently non-metric

We argue that the non-metric considerations are inherent to the protein sequence space, and should not be overlooked due to arising computational difficulties. Some of this difficulty stems from limited detection of highly divergent sequences by sequence similarity patterns across large families (clusters). This information is overlooked by single-linkage methods, which only use $O(N)$ of the input edges.
by a systematic scan for putative evolutionary links in the novel tree, most of them have been previously overlooked (submitted for publication).

**ACKNOWLEDGEMENTS**

We thank the CluSTr group at EBI for providing their data, and especially Anthony Quinn for his kind help. This article has also gained inspiration from discussions with Nati Linial, Ori Sasson, Roy Varshavsky and the ProtoNet group.

**Funding:** Y.L., E.P and M.F. are members of the SCCB, the Sudarsky Center for Computational Biology. The work is supported by the BioSapiens NoE (EU F6).

**Conflict of Interest:** none declared.

**REFERENCES**

Alicehof,S.F. et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**, 3389–3402.

Ashburner,M. et al. (2000) Gene ontology: tool for the unification of biology. The Gene ontology consortium. *Nat. Genet.* **25**, 25–29.

D’haeseleer,P. (2005) How does gene expression clustering work? *Nat. Biotechnol.*, **23**, 1496–1501.

Durbin,R. et al. (1999) Biological Sequence Analysis - Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, Cambridge, UK.

Finn,R. et al. (2006) Pfam: clans, web tools and services. *Nucleic Acids Res.*, **34**, D254–D255.

Fitch,W.M. and Margoliash,E. (1967) Construction of phylogenetic trees. *Science*, **155**, 279–284.

Frey,B.J. and Dueck,D. (2007) Clustering by passing messages between data points. *Science*, **315**, 972–976.

Kaplan,N. et al. (2004) A functional hierarchical organization of the protein sequence space. *BMC Bioinformatics*, **5**, 196.

Kaplan,N. et al. (2005) ProteNet 4.0: a hierarchical classification of one million protein sequences. *Nucleic Acids Res.*, **33**Database issue), D216–D218.

Kifer,I. et al. (2005) Predicting fold novelty based on ProteNet hierarchical classification *Bioinformatics*, **21**, 1020–1027.

Krause,A. et al. (2005) Large scale hierarchical clustering of protein sequences. *BMC Bioinformatics*, **6**, 6–15.

Lazareva-Ulitskaya,B. et al. (2003) On the quality of tree-based protein classification. *Bioinformatics*, **19**, 1876–1890.

Lin,J. and Rost,B. (2003) Domains, motifs and clusters in the protein universe. *Curr. Opin. Chem. Biol.*, **7**, 5–11.

Maiden,M.N.C. et al. (2007) New developments in the interplo database. *Nucleic Acids Res.*, **35**suppl. 1), D224–D228.

Murzin,A.G. et al. (1995) SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, **247**, 536–546.

Petrezyk,R. et al. (2005) The predictive power of the CluSTr database. *Bioinformatics*, **21**, 3604–3609.

Saxonson,O. et al. (2006) Functional annotation prediction: all for one and one for all. *Protein Sci.*, **15**, 1557–1562.

Shachar,D. and Linial,M. (2004) A robust method to detect structural and functional remote homologues. *Proteins*, **57**, 531–538.

Sriram,PH. (1957) The application of computers to taxonomy. *J. Gen. Microbiol.*, **17**, 201–226.

Sokal,R.R. and Michener,C.D. (1958) A statistical method for evaluating systematic investigation of sequences and structures. *J. Mol. Biol.*, **23**, 536–540.

Tatusov,R.L. et al. (1997) A genomic perspective on protein families. *Science*, **278**, 631–637.