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

Yaniv Loewenstein[1,*], Elon Portugaly[1], Menachem Fromer[1] and Michal Linial[2,*]

[1]School of Computer Science and Engineering and [2]Department of Biological Chemistry, Institute of Life Sciences, The Hebrew University of Jerusalem, Israel

ABSTRACT

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.

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.

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, ProtoNet4 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.

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.

Contact: lonshy@cs.huji.ac.il

1 INTRODUCTION

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 connections are especially evident in the biological domain. Gene Ontology (GO; Ashburner *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 *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 *et al.*, 2007).

ProtoNet (Kaplan *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 *et al.*, 2006), remote homology (Shachar and Linial, 2004) and structure (Kifer *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 *et al.*, 2005) and CluSTr (Petryszak *et al.*, 2005), which utilize single-linkage clustering to cope with the data size.

1.2 Hierarchical clustering by UPGMA

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)	$|E|$ (No. of edges)
Current – UniRef90 rel. 8.5, June 2006 ($N=1.80$M)		
Raw BLAST similarities (directed multi-graph)	50	2.5×10^9
Actual sparse similarities (symmetric, undirected)	30	1.5×10^9
Full non-sparse possible similarities	12 981	1.6×10^{12}
Extrapolated Future – UniRef90 rel. 13.1, March 2008 ($N=3.61$M)		
Actual sparse similarities (symmetric, undirected)	122	6.1×10^9
Full non-sparse possible similarities	52 262	6.5×10^{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 40–200 M edges.

Fig. 1. Illustration of non-metric constraints imposed by BLAST similarities (edges) for three multidomain Swiss-Prot sequences. False transitivity is possible due to CSKP_HUMAN. The missing triangle edge is due to no biological relatedness and no sequence similarity. It violates the triangle inequality since $e_{jl} \not\leq e_{il} + e_{ij}$. Triangle-offending missing edges prevent UPGMA from falsely clustering unrelated proteins together, by increasing average cluster dissimilarities.

methods, use the nearest pair of data points. A careful single-linkage implementation can cluster the whole set, reading one edge at a time ($O(1)$ edges in memory). Hence, single linkage is scalable to large datasets, however it is highly susceptible to outliers since only the minimum edge is considered in each step.
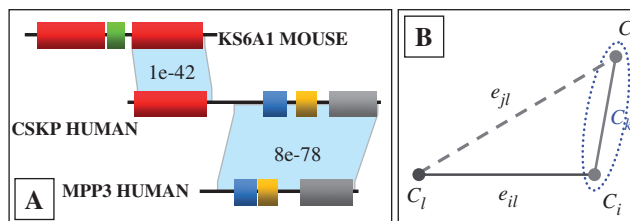
The most commonly used hierarchical clustering formulation is average linkage (Sokal and Michener, 1958) commonly referred to as UPGMA . It uses the mean similarity across all cluster data points and is thus more robust than single-linkage methods. UPGMA is intuitively appealing, and is a particularly practical algorithm owing to the stability of the arithmetic mean. It has been cited extensively, especially in the biological domain (e.g. D'haeseleer, 2005).

UPGMA is a text-book algorithm for correct reconstruction of sequence divergence processes (Durbin et al., 1999). Thus the exact UPGMA solution carries practical implications for understanding of tree processes underlying molecular evolution. Whilst more sophisticated algorithms exist for reconstruction of evolutionary trees, UPGMA 's runtime scales well to input size, and is often the algorithm of choice for the clustering of large sets with comparable performance (Lazareva-Ulitsky et al., 2005).

The problem however, is that UPGMA (or e.g. complete-linkage) requires the entire similarity matrix in memory. Since the number of pairwise relations is asymptotically quadratic in the number of clustered items ($O(N^2)$), the matrix becomes impossibly large to hold in memory even for moderately large datasets. Even for sparse similarity data, as in the case of a BLAST (Altschul et al., 1997) similarities reported here, data is still orders of magnitude too large to hold in memory for UPGMA. Due to rapid accumulation of data (e.g. Table 1), UPGMA is rendered impractical for a variety of tasks.

Current large-scale methods for sequence clustering cope with the size issue by either single-linkage clustering (Krause et al., 2005; Petryszak et al., 2005), supervised data pre-selection (e.g. Tatusov et al., 1997), or other heuristics. The ProtoNet method (Kaplan et al., 2005) addressed size by using a reduced set of Swiss-Prot sequences (10%) to build a high quality UPGMA tree skeleton, to which the larger TrEMBL set was appended.

Here, we develop a framework for finding the correct UPGMA tree for very large data under any practical memory constraint. The mass of all detectable sequence similarities in the entire set of protein sequences is now directly clusterable. The result is a

robust evolutionary-driven tree of nearly 2 million non-redundant sequences. The tree corresponds very well with known protein families.

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.

2 THEORY

2.1 Preliminaries

Agglomerative hierarchical clustering algorithms differ in the choice of cluster dissimilarity objective. For clarity, we refer only to UPGMA (average-linkage) throughout the article. The concepts presented here, however, easily apply to other hierarchical clustering formulations as well.

2.1.1 Problem statement and notation UPGMA takes an input undirected graph of $G=(\mathbb{V},\mathbb{E})$ and edge weights d, where $\mathbb{V}$ is the set of data items (vertices), $\mathbb{E} \subset \mathbb{V} \times \mathbb{V}$ is the edge set, and $d: \mathbb{E} \to \mathbb{R}^+$ denotes the actual pairwise dissimilarities. We define $N \overset{\Delta}{=} |\mathbb{V}|$.

As clusters are being merged, multiple edges are possible between a pair of non-singleton clusters. These are collated into average edges, and are referred to as '*thick*' edges throughout this article. Thick edges are cluster-pair unique, and possibly encompass more than one '*thin*' input edge in the cluster multi-graph. For presentation clarity, we will assume that $\mathbb{E}$ is symmetric (i.e. $d_{ij}=d_{ji}$) and ignore meaningless self-loops (e_{ii}). We will think of dissimilarities as distances. Hence lower dissimilarities can be interpreted as 'closer' clusters. However, we do not require that dissimilarities are metric distances, by the formal mathematical notion, unless otherwise stated. Moreover, the triangle inequality does not hold for our data (Fig. 1).

2.2 Sparse UPGMA

$\mathbb{E}$ is considered *sparse* if not all possible pairs exist in the input graph ($\exists i, j \in \mathbb{V} : e_{ij} \notin \mathbb{E}$). For instance, the sets ($\mathbb{E}$) 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.

$$d_{ij} \stackrel{\Delta}{=} \frac{1}{|C_i||C_j|} \sum_{p \in C_i, q \in C_j} d_{pq} \qquad (1)$$

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

$$\psi \stackrel{\Delta}{=} \max(\mathbb{E})$$

$$d_{ij} \stackrel{\Delta}{=} \begin{cases} d_{ij} & \text{if } e_{ij} \in \mathbb{E} \\ \psi & \text{otherwise.} \end{cases}$$

Where input d_{ij}'s are used when available and ψ is used otherwise. Hence, ψ 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 for missing value completion across connected clusters. Throughout this work, e_{ij}'s are replaced with d_{ij} in Equation (1) to denote the actual average dissimilarity.

2.2.2 Leveraging sparsity
For suitable inputs, a sparse representation for $\mathbb{E}$ decreases the memory requirement considerably. Furthermore, new thick edges can efficiently be computed recursively (in $O(1)$). This calculation, denoted as $\stackrel{\psi}{|C_*|}\mathcal{F}(\bullet, \bullet)$ hereby, requires ψ, cluster sizes $|C_*|$, and two precalculated edges. For UPGMA we have:

$$d_{kl} \stackrel{C_k = C_i \cup C_j}{=} \stackrel{\psi}{|C_*|} \mathcal{F}_{\text{UPGMA}}(d_{il}, d_{jl}) = \frac{|C_i| d_{il} + |C_j| d_{jl}}{|C_i| + |C_j|} \qquad (2)$$

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 $\mathbb{E}$ sorted, *Sparse-UPGMA* requires $O(E \lg V)$-time and $O(E)$-memory, instead of the straightforward $O(V^3)$-time and $O(V^2)$-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 ProtoNet4 data (114K) requiring time (3 min on a 2.80 GHz machine) that was negligible as compared to the similarity

Fig. 2. Sparse UPGMA. E is typically implemented as a minimum heap data structure, extended with logarithmic time removal of non-minimum elements. The output tree and cluster heights are given by e_{ij}s. The algorithm is correct for non-sparse inputs as well.

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 $\mathbb{E}$ 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, $\mathbb{E}^t$, between valid (unmerged) clusters. Initially, $\mathbb{E}^0 \stackrel{\Delta}{=} \mathbb{E}$. 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 $\mathbb{E}_\lambda^t$. Higher Ms may require fewer rounds (due to additional progress per round). We also define

$$\lambda^t \stackrel{\Delta}{=} \text{ maximum of (at most } M \text{) loaded edges at round } t$$
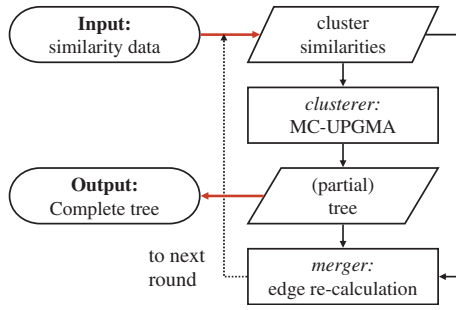
Fig. 3. Multi-round MC-UPGMA scheme.

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 $\psi = 100$. Let C_i and C_j be merged to some $C_k = C_i \cup C_j$ at height $d_{ij} = 1$. Let $d_{il} = 2$, and $d_{jl} \geq \lambda$ (a non-loaded edge). Suppose C_k is merged with some $C_r \neq C_l$ at height $d_{kr} = 7$. Then, C_r may not have been the closest neighbor, if e.g. $d_{jl} = 10$ (since $d_{kl} = \frac{1*2+1*10}{1+1} = 6 < d_{kr} = 7$). On the other hand, if C_k and C_l are indeed merged, then set $d_{jl} = \psi = 100$ and thus merging is still incorrect ($d_{kr} = 7 < d_{kl} = \frac{1*100+1*2}{1+1} = 51$).

Hence, a correct clusterer should be mindful of unseen edges ($\geq \lambda$), effecting clustering before λ. Such examples are rather prevalent in non-metric datasets. Figure 1 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 of d_{ij} is lower (l_{ij}) and upper (u_{ij}) bounded as follows:

$$l_{ij}^t \triangleq \begin{cases} d_{ij} & \text{if } e_{ij} \in \mathbb{E}_\lambda^t \\ \lambda^t & \text{otherwise.} \end{cases}$$

$$u_{ij}^t \triangleq \begin{cases} d_{ij} & \text{if } e_{ij} \in \mathbb{E}_\lambda^t \\ \psi & \text{otherwise.} \end{cases}$$

It follows that $l_{ij} \leq d_{ij} \leq u_{ij}$ (for clarity, round indices are omitted hereafter). Hence, maintained edges e_{ij} are now represented as uncertainty intervals $[l_{ij}, u_{ij}]$, rather than by exact d_{ij} values. The exact value of d_{ij} is not known to the clusterer unless $u_{ij} = l_{ij}$.

2.4.5 Clusterer algorithm The clustering algorithm (Fig. 4) design is rather similar to *Sparse-UPGMA*. Now however, the clusterer loads only the M minimal edges in $\mathbb{E}^t$ (which determine λ^t). Edges d_{ij} are replaced with intervals $[l_{ij}, u_{ij}]$ to accomodate uncertain edge values (due to partial edge data at hand). Clustering halts when it is impossible to identify the minimal edge in the entire $\mathbb{E}^t$, including the edges in $\mathbb{E}^t \setminus \mathbb{E}_\lambda^t$, which are unknown to the clusterer.

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_λ^t or (B) the minimal edge may be at hand but could not be determined unequivocally (as in the given counter example). Since unseen edges are $\geq \lambda$, requiring that a merged edge e_{ij} satisfies $u_{ij} \leq \lambda$ assures that situation (A) never happens. The latter

Fig. 4. Multi-round MC-UPGMA clustering—the clusterer. This is a modified version of the code for *Sparse-UPGMA* (Fig. 2).

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— an edge whose upper bound u_{ij} is below the lower bound of any other edge in $\mathbb{E}_\lambda^t$ (i.e. $\exists e_{ij} \in E : \forall e_{rs} \in E : u_{ij} \leq l_{rs}$; Fig. 4). By maintaining the criterion $u_{ij} \geq l_{rs}$ in the case where $(i,j) = (r,s)$ 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 $|\mathbb{E}^{t+1}|$ is accordingly reduced at a quadratic rate. Once all edges fit in memory ($|\mathbb{E}^T| \leq M$), 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, than Multi-Round MC-UPGMA clusters all edges $\leq \frac{\lambda}{2}$

PROOF. Exact minimal edges do not halt clustering. We will show that inexact edges appear only after $\frac{\lambda}{2}$. Let $e_{kl} \in \mathbb{E}_\lambda^T$ be an inexact edge appearing along the clustering process, and assume w.l.o.g.

that C_k was created before C_l is merged. Let C_i and C_j be the clusters merged to create C_k, i.e. $C_k = C_i \cup C_j$. Either $d_{il} \leq \lambda$ or $d_{jl} \leq \lambda$, otherwise $d_{kl} \geq \lambda \Rightarrow e_{kl} \notin \mathbb{E}_\lambda^t$. Assume w.l.o.g. that only $d_{il} \leq \lambda$ (i.e. $d_{jl} \not\leq \lambda$), since otherwise e_{kl} would not have been inexact. From merge order, we have $d_{ij} \leq d_{il} \leq \lambda$. Now, note that $d_{il} \geq \frac{\lambda}{2}$, otherwise $d_{ij} \leq d_{il} < \frac{\lambda}{2}$ implying $d_{jl} \leq d_{il} + d_{ij} < \frac{\lambda}{2} + \frac{\lambda}{2} = \lambda$ (due to the triangle inequality), contradicting our assumption. Plugging in Equation (2), we have $e_{kl} = \frac{e_{il}|C_i| + e_{jl}|C_j|}{|C_i||C_j|} \geq \frac{\frac{\lambda}{2}|C_i| + \lambda|C_j|}{|C_i||C_j|} \geq \frac{\lambda}{2}$. ∎

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 E-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 ψ 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 $e_{ij} \leq (1 + \epsilon)(e_{ik} + e_{kj})$ is satisfied for some fixed $\epsilon > 0$.

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 LOADEDGES, Fig. 5). Merging requires the previous set of edges $\mathbb{E}^{t-1}$ and the forming tree. A naive algorithm will form $\mathbb{E}^t$ 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 $\mathbb{E}^t \setminus \mathbb{E}^{t-1}$. 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 λ-missing edges and uncertainty intervals. Clearly, when *Multi-Round MC-UPGMA* halts, it is not using its entire memory budget M, since each merge reduces the number of edges in memory. The newly formulated hybrid algorithm


```
Procedure Single-Round-MC-UPGMA (E,M,ψ)
  ... //initialization
  while(LOADEDGES(E,M,fh,parents) = TRUE) do
      λ ← max E
      while(∃e_ij ∈ E : ∀e_rs ∈ E : ũ_ij ≤ l̃_rs ≤ λ) do
          ... // as in Sparse-UPGMA
          parents[i] ← parents[j] ← k
                              l≠j          l≠i
          foreach C_l ∈ {C_l : (e_il ∈ E) ∨ (e_jl ∈ E)} do
              d̃_kl ← d̃_il ⊕ d̃_jl
              E ← E ∪ {e_kl} \ {e_il, e_jl}
          end foreach
      end while
      if (|E| = M) then // ⇒still stuck after reloading
          LOOKAHEAD(E,fh,ψ,parents) //⇒only edges exact
      end if
  end while

Procedure LOADEDGES(E,fh,M,parents,|C_*|)
```

Input: E - current edge set, fh - edge file handle, M - memory constraint, parents - cluster ancestors, $|C_*|$ - cluster sizes
Output: returns TRUE if edges remain on disk, FALSE otherwise. E and $\tilde{d}$ are modified.

```
while(|E| < M) do
    if(¬HasNext(fh)) then return FALSE //No edges remain
    e_ij = Next(fh)
    p_i ← FINDANCESTOR(i,parents)
    p_j ← FINDANCESTOR(j,parents)
    if (p_i ≠ p_j) then
        d̃_{p_i p_j} ← d̃_{p_i p_j} ⊕ < e_ij, |C_i||C_j| >
        E ← E ∪ {e_ij}
    end if
end while
return TRUE //some edges remain
```

Fig. 5. Illustration of *single-round MC-UPGMA*. FINDANCESTOR uses a suitable data structure for disjoint sets to allow efficient parent look-up. LOOKAHEAD peeks at unloaded edges on disk, and loads only components of currently maintained edges. Afterwards, uncertain edges can be inferred as definitively missing. See text for definition of $\tilde{d}$ and $\oplus$ notations.

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 λ increases dynamically and is no longer fixed. Hence, λ-dependent recalculation of l_{ij}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.

This capability is due to the introduction of an update procedure for edge intervals.

2.6.3 Edge representation and notation Let $\widetilde{d}_{ij} \in \mathbb{R}^+ \times \mathbb{N}$ i.e. $\widetilde{d}_{ij}$ is now a pair of two quantities; (1) an unnormalized sum of seen singleton (thin) edges and (2) a count parameter, a non-negative integer indicating the number of seen edges accounted for in the sum. We denote the first component as $edge\_sum(\widetilde{d}_{ij})$ and the second as $count(\widetilde{d}_{ij})$, corresponding to the numerator and denominator in Equation(2), respectively. We define the $\oplus$ operator for $\widetilde{d}_{ij}$ as the piecewise (dot) plus operator.

The missing e_{il} (e_{jl}), previously replaced with a fixed λ bound, is now dynamically updated with the current λ-value:

$$\widetilde{l}_{ij} \overset{\triangle}{=} \frac{edge\_sum(\widetilde{d}_{ij}) + \lambda\left(|C_i||C_j| - count(\widetilde{d}_{ij})\right)}{|C_i||C_j|} \quad (3)$$

$$\widetilde{u}_{ij} \overset{\triangle}{=} \frac{edge\_sum(\widetilde{d}_{ij}) + \psi\left(|C_i||C_j| - count(\widetilde{d}_{ij})\right)}{|C_i||C_j|} \quad (4)$$

We think of $\left(|C_i||C_j| - count(\widetilde{d}_{ij})\right)$ as a dynamic uncertainty weight. Now, when a missing component of an uncertain edge is read while reloading edges from disk, $count(\widetilde{d}_{ij})$ is incremented. Consequently the respective uncertainty weight diminishes, and the interval tightens up around d_{ij}. Fully linked edges (i.e. $count(\widetilde{d}_{ij}) = |C_i||C_j|$) 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, λ grows as more edges are loaded. Because uncertain intervals are computed dynamically, lower bounds become tighter as λ 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 $N \ll M$) 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 (Liu 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 $E = 100$ 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 transformed to an undirected graph (symmetric dissimilarities) by keeping only $e_{ij} = \min(e_{ij}, e_{ji})$ for $i < j$, i.e. half of the now-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 keywords

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, with a total of 3752 (3528 are ≥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_i, a protein in C_i 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.

$$\mathcal{J}(C_i, k) \overset{\triangle}{=} \frac{|C_i \cap k|}{|C_i \cup k|} = \frac{TP}{TP+FP+FN} \quad (5)$$

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 $\mathcal{J} \leq$ specficity and $\mathcal{J} \leq$ 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

$$\mathcal{J}(k) \triangleq \max_{C_i \in \mathbb{C}: |C_i| > 1} \mathcal{J}(C_i, k) \qquad (6)$$

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 $\mathcal{J}^w$ a family-size-weighted average (protein-centric).

3.6 Comparison with other methods

To evaluate the contribution of the newly formulated UPMGA 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 $E = 1e-40$ than we ($E = 100$). CluSTr Slim is a pruned tree of clusters ≤ 1000 with <90% cluster overlap, 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 only 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.

4.1 The complete protein families tree

The process started from 1 801 506 UniRef90 proteins. 1107 (0.06%) proteins are singletons having no BLAST similarities, and do not enter the clustering process. From the clustered set, 1 791 206 proteins (99.5%) are clustered into a single tree.

For the first time, we are able to provide an extensive tree of all UniProt sequences in a suitable evolutionary context. The tree is built from the massive set of all sequence similarities detected by an exhaustive search comparing all non-redundant sequences in the UniRef90 set, and agrees very well with external resources. Multiple nearly identical UniProt sequences may be collapsed into a single UniRef90 representative. Here we analyze the UniRef90 collapsed tree, to avoid redundancy issues due to overrepresented protein families. Since our method is unsupervised, it is not fine-tuned for detection of well-characterized protein families.

4.2 MC-UPGMA maintains performance on hard data

The performance of MC-UPGMA on different sets is summarized in Table 2. Swiss-Prot (220 K) is a relatively easy set for UPGMA as well as for single linkage, and surprisingly the performance relative to InterPro families is quite high. Expanding the analysis to a much larger set (UniRef90) slightly reduced the performance for MC-UPGMA but gravely affected single linkage. This observation suggests that Swiss-Prot does not represent the complexity of the whole sequence space. On the other hand, the performance on the UniRef50 set is surprisingly stable (relative to UniRef90). Considering the significantly increased difficulty of UniRef50, this suggests that the richness of the entire tree is maintained by this set. The improvement of MC-UPGMA over single linkage is even more dramatic for the harder case of InterPro domains. Furthermore, UPGMA demonstrated enhanced capacity for handling large protein families (Table 3 and Figure 6). Single linkage performs poorly for large families and thus performance is deteriorated by family size weighting. The ProtoNet4 protocol performs fairly well for large families, since large families are presumably well represented in the Swiss-Prot skeleton. The improvement is due to enhanced sensitivity (Fig. 6).

Notably, the 1.54 M clusters provided by *CluSTr slim* produced significantly lower performance than the 1.17 M single-linkage clusters (tied merges are pooled to non-binary clusters). Single-linkage tree performance suffered only minor performance loss, when the CluSTr SLim pruning protocol was applied (not shown). Hence, the difference in performance does not stem from tree pruning. Furthermore, the CluSTr Slim data contained clusters at a variety of granularities. Thus, it is hard to establish the reason for this discrepancy, considering that the two methods do not use exactly the same data.

4.3 Sparse sequence space—well-connected tree

Only 0.09% of the edges (pairs related by BLAST) exist in the input sequence similarity graph (Table 1). However, 1 497 733 of the tree clusters (83.6%) are fully linked, including 426 360 large clusters with at least 10 members. Hence, the edge distribution in the resulting tree is highly non-random even though a very permissive BLAST cutoff was used. Edges appear in dense clusters corresponding to protein families at different levels of evolutionary granularities. The surprisingly high connectivity of the protein tree emphasizes the potential wealth of information laying in

Table 2. Clustering performance evaluation based on InterPro keywords

	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
UniProt						
CluSTr Slim	0.470	0.955	0.489	0.375	0.889	0.406

UniRef90 (1.8M sequences) reflects non-trivial difficulty (sequence identity <90%). UniRef50 (960K) reflects a hard setting (<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 ($\mathcal{J}^w$)

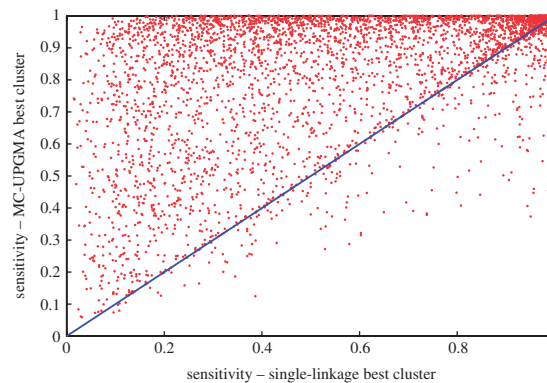|	InterPro Families		InterPro Domains	
	$\mathcal{J}$	$\mathcal{J}^w$	$\mathcal{J}$	$\mathcal{J}^w$
MC-UPGMA (current)	0.900	0.856	0.735	0.654
Single-linkage	0.808	0.598	0.566	0.306
ProtoNet4 protocol	0.795	0.782	0.669	0.618

Table 4. Clustering progress for the hard UniRef90 data

t	No. of merges	Last merge	|tree|	$\|E^t\|$	$\|E^t\|/\|E^0\|$ (%)
1	679915	6e−69	679 915	5.58e+08	36.77
2	430486	1e−24	1 110 401	3.15e+08	20.77
3	258122	8e−05	1 368 523	2.24e+08	14.78
4	26774	0.004	1 395 297	2.14e+08	14.06
5	39712	0.444	1 435 009	1.89e+08	12.47
6	11534	1.052	1 446 543	1.82e+08	11.98
7	5225	1.474	1 451 768	1.78e+08	11.75
8	5948	2.062	1 457 716	1.75e+08	11.50
9	11011	3.533	1 468 727	1.68e+08	11.03
10	4286	4.277	1 473 013	1.65e+08	10.88

First 10 iterations are shown out of total 200. Here, clustering progress per round is significantly slowed down in advanced iterations (see text).

similarity patterns across large families (clusters). This information is overlooked by single-linkage methods, which only use $O(N)$ of the input edges.

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 ($x = y$). Points above diagonal correspond to higher sensitivity for MC-UPGMA and vice versa. Average specificities are comparable across this set (MC-UPGMA = 0.90 ± 0.15, single-linkage = 0.88 ± 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.
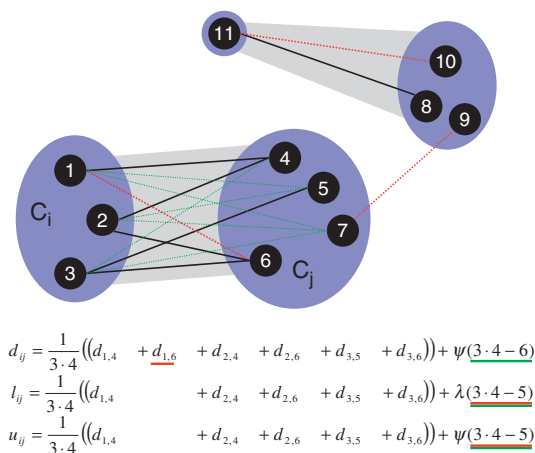
4.5 Analysis of multi-round MC-UPGMA run

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

$$d_{ij} = \frac{1}{3\cdot 4}\left(\left(d_{1,4} + d_{1,6} + d_{2,4} + d_{2,6} + d_{3,5} + d_{3,6}\right)\right) + \psi(3\cdot 4 - 6)$$

$$l_{ij} = \frac{1}{3\cdot 4}\left(\left(d_{1,4} + d_{2,4} + d_{2,6} + d_{3,5} + d_{3,6}\right)\right) + \lambda(3\cdot 4 - 5)$$

$$u_{ij} = \frac{1}{3\cdot 4}\left(\left(d_{1,4} + d_{2,4} + d_{2,6} + d_{3,5} + d_{3,6}\right)\right) + \psi(3\cdot 4 - 5)$$

Fig. 7. Illustration of thick edges (grey) connecting clusters (blue), and thin input edges (red and black). A thick edge calculation for a pair of clusters C_i and C_j is demonstrated based on three possible components: known ($\leq \lambda$) thin edges (black), unknown thin edges (due to the memory constraint) that do exist (red, dashed lines), and possible edges connecting C_i and C_j members but do not exist in the sparse input graph (green). Corresponding components in the calculation are color coded accordingly. It is impossible to distinguish non-existing edges from unknown edges without reading the whole input graph. The cluster graph is broken into two connected components in $\mathbb{E}_{\lambda}^{t}$, as long as $d_{7,9}$ is missing.

alignment, as reflected by the partial connectivity of most protein families. However, non-metric constraints are inherent to the protein sequence space due to the modular nature of protein domains. Domain shuffling events may create mixtures of distinct divergence processes, which could not be captured by a single tree for whole proteins. Yet, incorporating this constraint into the clustering problem—by taking into account missing edges in the average cluster dissimilarity—seems to aid UPGMA avoid the pitfalls of false transitivity, as reflected by local sequence alignment (Fig. 1). This is demonstrated by the significant improvement of UPGMA over single-linkage methods as shown for the case of InterPro protein domains (Table 2). Even when the reduced set of Swiss-Prot proteins is used to partition the much larger UniRef90 set (ProtoNet4 in Table 2), the underrepresented UPGMA skeleton still significantly outperforms single-linkage clustering using the whole data.

4.7 Top levels of tree surprisingly meaningful

Since a significant amount of protein families coincide with poorly connected clusters, we stress that the top levels of the UPGMA cluster tree are biologically meaningful and should not be pruned out. Furthermore, our results show that hidden, remote connections between protein families, some overlooked by state-of-the-art specialist methods (e.g. Profile–Profile comparisons), are picked up at these seemingly uninteresting levels. Averaging across large sets of sequence similarities is able to weed out these faint connections, otherwise below the level of random BLAST hits. Indeed, 710 evolutionary connections between protein families were suggested

by a systematic scan for putative evolutionary links in the novel tree, most of them have been previously overlooked (submitted for publication).

REFERENCES

Altschul,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**, 1499–1501.

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**, D247–D251.

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) ProtoNet 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 ProtoNet hierarchical classification. *Bioinformatics*, **21**, 1020–1027.

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

Lazareva-Ulitsky,B. *et al.* (2005) On the quality of tree-based protein classification. *Bioinformatics*, **21**, 1876–1890.

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

Mulder,N.J. *et al.* (2007) New developments in the interpro 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**, 536–540.

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

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

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

Sneath,P.H. (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 relationships. *Univ. Kans. Sci. Bull.*, **28**, 1409–1438.

Suzek,B.E. *et al.* (2007) UniRef: comprehensive and non-redundant UniProt reference clusters. *Bioinformatics*, **23**, 1282–1288.

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