Protein Classification Based on Propagation on Unrooted Binary Trees

András Kocsor[1], Róbert Busa-Fekete[1] and Sándor Pongor[2,3,*]

[1]*Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and University of Szeged, H-6720, Szeged, Aradi vértanúk tere 1., Hungary;* [2]*Protein Structure and Bioinformatics Group, International Centre for Genetic Engineering and Biotechnology, Padriciano 99, I-34012 Trieste, Italy;* [3]*Bioinformatics Group, Biological Research Centre, Hungarian Academy of Sciences, Temesvári krt. 62, H-6701 Szeged, Hungary*

Abstract: We present two efficient network propagation algorithms that operate on a binary tree, i.e., a sparse-edged substitute of an entire similarity network. TreeProp-N is based on passing increments between nodes while TreeProp-E employs propagation to the edges of the tree. Both algorithms improve protein classification efficiency.

Keywords: Propagation algorithm, PageRank, protein classification, ROC analysis, phylogenomics.

1. INTRODUCTION

Propagation algorithms have gained importance in several areas of pattern recognition and information retrieval. Belief propagation or message passing [1,2], the *PageRank* algorithm [3] and the power method [4] are all based on propagating information through a network that consists of nodes and edges. Several propagation algorithms were successfully used in practice, the PageRank algorithm used by the Google WEB surfer perhaps being the best known example.

Protein classification is a crucial task in genome annotation, and virtually all major algorithms of machine learning have been applied to this problem with varying success [5-7]. Propagation algorithms were also successfully employed in this field [8-10]. The apparent advantage of this approach originates from the fact that classification does not just rely on a simple similarity measure but also on the entire network protein similarities. In fact, the first application of a *PageRank* style algorithm to protein classification [8] demonstrated that propagation yields a definite increase in classification efficiency. On the other hand, protein similarity networks are large, and they typically contain several hundred thousand proteins as nodes and several million pairwise similarity values as links, which makes propagation rather time-consuming. To overcome this, one possibility is to use smaller networks for the propagation, like bipartite graphs [9] or threshold graphs [11]. In this paper we explore another avenue that instead of propagating on the entire network focuses on replacing a relevant part of the protein similarity network by a structured description. This approach is based on the well known observation that protein similarity networks are modular, i.e. they contain densely connected subgraphs around each protein class, wherein the biologically important similarities between members of the same class can be easily distinguished from the accidental similarities that exist between members of different classes [12,13]. In addition, structured descriptions such as phylogenetic trees are known to capture the important features of a classification scheme, so they may serve as a natural noise reduction filter for propagation algorithms. Here we propose to replace (a selected part of) the protein similarity network with a binary tree on which the propagation will be carried out. Binary trees are sparse structures as compared to a full network of similarities, so there is potential gain in speed. On the other hand, one has to build a binary tree, which is an extra burden, but as we will see, there is a substantial overall gain in computing time.

We will now describe two heuristic protein classification algorithms that use propagation on binary trees. *TreeProp-N* is modeled on the *PageRank/RankProp* philosophy. It uses an initial ranking vector based on the distance of the database entries from the query measured along the edges of the tree. This ranking is then updated by propagating information along the edges of the tree. *TreeProp-E* uses a different propagation principle: the initial data is stored in the weighted edges of the binary tree, and the weights afterwards are propagated to the neighboring edges by a simple update rule. We will use the fast tree-building algorithm FastME [14] to construct the initial tree for both algorithms.

The rest of this paper is organized as follows. In the next section we will introduce the *PageRank* method and its applications to protein classification tasks (Section 2). Then we will introduce two new algorithms called *TreeProp-N* and *TreeProp-E* that use phylogenetic trees for the propagation (Section 3 and Section 4). The efficiency of our methods will be compared on various real-life benchmark datasets, including protein sequences as well as protein 3D structures (Section 5). Section 6 contains a summary of the results and a discussion. All the other algorithms and datasets used here are described in the Appendix section.

2. *PAGERANK* AND ITS APPLICATION TO PROTEIN CLASSIFICATION

Originally, the PageRank algorithm [3] was developed for information retrieval purposes. There are many other

*Address correspondence to this author at the Protein Structure and Bioinformatics Group, International Centre for Genetic Engineering and Biotechnology, Padriciano 99, I-34012 Trieste, Italy and Bioinformatics Group, Biological Research Centre, Hungarian Academy of Sciences, Temesvári krt. 62, H-6701 Szeged, Hungary Tel: +36 62 546713; Fax:+36 62 645508; E-mail: pongor@icgeb.org

0929-8665/08 $55.00+.00

areas where this simple idea was adopted with success, including Natural Language Processing [15], Word Sense Disambiguation [16] and Protein Classification [8]. The common representation underlying these diverse fields is an a priori known similarity network of objects (i.e. a weighted graph, where the points correspond to the objects, and the edges represent the similarities among them). Each similarity network can be represented as a stochastic matrix S in which each row sums up to one and which has no negative entry. An entry of the matrix S represents the similarities between two objects. PageRank converges to the stationary point of transformation S (i.e., the stationary distribution of those Markov chains where the transition matrix is S). We can express this ranking in an iterative fashion using the following update rule:

$$y(t+1) = Sy(t) \tag{1}$$

where *y(t+1)* denotes the similarity score after t iterations. Since this rule corresponds to the ordinary power method [17], the convergence is fulfilled [4]. Moreover the *Y={y(0),y(1),...,y(T),...}* sequence will converge to the biggest eigenvector of the stochastic matrix *S*, which is - due to the Perron-Frobenius theorem - equal to 1. We should note here that this method computes a single rank value for each object of the given network, i.e. this is not yet a query-based algorithm. Besides this the convergence of this process is known to depend on the gap between the first and second eigenvalues [17]. It may thus be worthwhile to explore the eigenvalue difference in order to estimate the necessary number of iterations before we use this approach on a similarity network. With biologists it is normal practice to have an unknown protein sequence which is compared to a database using a sequence comparison tool, like Smith-Waterman (SW) algorithm [18] or BLAST algorithms [19]. The result is a ranking of the database entries according to a similarity score with respect to the query, and the top hits are evaluated either by an expert or by automated procedures. The query-based or personalized version of PageRank is more applicable for this type of a problem. Let us denote the protein entries of the database by $p_1, p_2, ..., p_N$ and the query protein by q. Now let's define a pairwise similarity measure between the *ith* and *jth* protein as $s(p_i, p_j)$ (typically this will be a BLAST or Smith Waterman similarity score). We can then organize the $s(p_i, p_j)$ similarities into a matrix *S*, where the rows will be normalized so that their sum equals 1. Then, with the help of matrix *S*, we can formulate the update rule of the so-called *personalized PageRank* method as

$$y(t+1) = y(0) + \alpha Sy(t) \tag{2}$$

Here *y(0)* is the initial vector of similarity scores and the α parameter is a constant in the range *[0,1]*. This process converges to the y^* fix-point of Eq. 2, and it can also be calculated analytically by solving the $(I-\alpha S)^{-1} y(0) = y^*$ system of linear equations. But the size of matrix *S* is *N*×*N*, so the analytical solution requires $O(N^3)$ time. On the other hand, we can adequately approximate y^* by *y*(*T*) (i.e. the *T*th element of the iteration sequence). The *personalized PageRank* algorithm was introduced for protein classification as the *RankProp* algorithm in the seminal paper of Weston *et al.* [8].

A slightly different update rule was also introduced by Zhou *et al.* in [11]:

$$y(t+1) = (1-\alpha)y(0) + \alpha Sy(t) \tag{3}$$

It is easy to show that the y^* limit point of Eq. 3 is equal to $(1-\alpha)(I-\alpha S)^{-1} y(0)$. This form of PageRank leads to the same ranking as Eq. 2.

As restricting the size of the similarity network is a plausible way of speeding up the calculation, bipartite graphs [9] or threshold graphs [11] were both used for this. In the following sections we will present two novel algorithms that approach the problem of size restriction via the use of binary (phylogenetic) trees instead of a similarity network.

3. *TREEPROP-N:* PROPAGATION ON THE NODES OF A BINARY TREE

The main idea behind the *personalized PageRank* method is the application of a diffusion operation on a network of pairwise protein similarity, according to Eqs. 2-3. In this section we will introduce an alternative algorithm, namely *TreeProp-N*, where a similar diffusion operation is applied on an unrooted weighted phylogenetic tree that was built up from a database and the query protein *q*. We should mention here that the phylogenetic tree is a binary tree graph wherein the leaves of the tree correspond to the biological objects. In our work, we used the FastME algorithm [14] to construct the tree from the pairwise similarity scores computed between the elements. If we use similarity scores $s_{ij} \in [0,1]$, then the tree will be built from a distance score expressed as $1 - s_{ij}$.

The propagation on the resulting tree in accordance with the update rule of the PageRank. The initial scores for each node (leaves and inner nodes) in the tree will be the shortest path between a given node and query element *q*. Since for the propagation we need similarity scores, rather than distance values we transform the shortest path values to similarity score $[0,1]$ by dividing them with the maximal shortest path and subtracting the result from 1.

The propagation is carried out on the tree *T*=(*V*,*E*), where |*V*|=*N*+*1*. An unrooted binary tree *T* with *N*+1 leaves has 2*N* nodes, so we can collect the shortest path from the *q* query point to the points of the *T* tree into a *y* vector of length *N*+1. According to the binary branching pattern of the phylogenetic tree, a leaf will only have one neighbor while an inner node will have three neighbors. Next we will denote the neighbors of a point *p* by *N*(*p*). The update rule can be written as:

$$y_i(t+1) = (1-\alpha)y_i(0) + \alpha \sum_{p \in N(p_i)} w(p, p_i) y_p(t), \tag{4}$$

where $w(p, p_i)$ stands for the weight of the edge between p and p_i within the tree, and $y_i(t)$ means the propagated value of the point p_i after the t iterations. The α parameter lies between [0,1] as it does in Eq. 3. This parameter sets the balance between the effect of tree-structure on the one hand, and the effect ranking on the other: higher values will emphasize the former over the latter.

The convergence of the propagation is ensured if the outgoing edge weights from a point sum up to one, which is provided by a normalization step. We should add that the resulting matrix has at most three elements in each of its rows, so the calculation can be carried out in $O(tn)$ time, where t is the number of iterations.

4. *TREEPROP-E*: PROPAGATION ON THE EDGES OF A BINARY TREE

In a conventional propagation algorithm, the nodes of a similarity networks send messages to each other, and the magnitude of the messages is then proportional to the similarity between the sender and receiver. Here we introduce a novel approach where the edge weights of a binary tree will be propagated. The idea behind this approach is similar to the original *PageRank* concept, because the weight of an edge will be determined by the weight of its neighbors.

In the first step we build up an $T=(V,E,w)$ unrooted weighted phylogenetic tree using query protein q and the known entries of a database. This tree has N+1 leaves and 2N-1 edges. Since we propagate on the edges the $y(i)$ vectors have a length of 2N-1. In a tree we will call two edges adjacent if they have a common endpoint. In an unrooted binary tree an edge pointing to a leaf has two adjacent edges, while an interior edge has four adjacent edges. Let us denote the set of adjacent edges with edge e by $N(e)$.

Now let $y(0)$ be the initial value of the edge lengths of the tree. Then we can apply the following simple propagation rule:

$$y_i(t+1) = (1-\alpha)y_i(0) + \alpha / \left|N(e_i)\right| \sum_{e_j \in N(e_i)} y_{e_j}(t) \qquad (5)$$

which means that the score of an edge in step (t+1) will depend on the mean value of the adjacent edge weights at step t as well as on its original weight at t=0. This update will be repeated a predetermined number of times, after which the ranking with respect to the query q can be calculated using the weighted path lengths between the query and the other proteins in the tree, calculated from the updated weights. The schematic overview of this method can be seen in Fig. (**1**).

The convergence proof of *PageRank* can be almost directly applied to *TreeProp-E*. If we rewrite the propagation rule in Eq. 5 into the matrix form, as in the case of PageRank (Eq. 3), we get a non-negative real matrix on which the Frobenius-theorem can be applied. And since its row sum is less than one, its spectral radius is also less than one so the convergence is ensured.

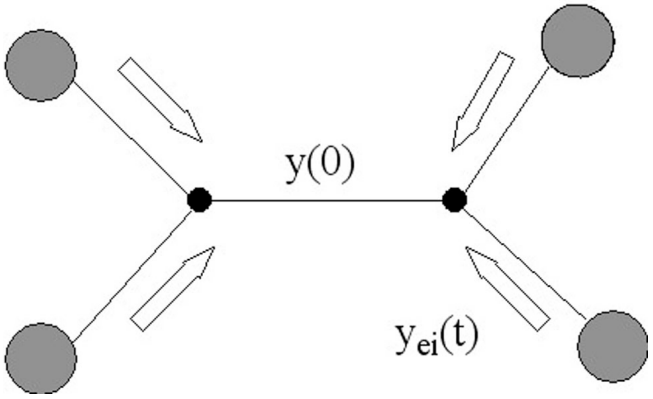

Figure 1. The process of the local operations on the edges. Each neighboring edges of an internal edge sends "messages" to its neighbor, and the magnitude of the message is proportional to the propagated edge weight.

5. EXPERIMENTS

5.1. Time Complexity and Practical Implementation

The personalized PageRank/RankProp method applies the propagation to an entire similarity network. For a network of N nodes this means a time-estimate of $O(iN^2)$ steps where i is the number of iterations. This can be time-consuming for large protein similarity networks that typically have several thousand to several hundred thousand nodes. The situation can be alleviated by considering just the first n objects nearest to the query along with all of their similarities, which leads to a time-estimate of only $O(in\mathrm{N})$, where i being the number of the iterations [8], [Supplementary data].

TreeProp-N and *TreeProp-E* apply the propagation to a binary tree. Since constructing the tree with the FastME algorithm requires $O(N^2)$ time in addition to propagation, it is necessary to reduce the size N of the input network. We propose a reduction where we consider just the m nearest neighbors for each of the n top-ranking objects, which will lead to a maximal time estimate of $O(inm)$. A phylogenetic tree is built from nm objects and it will have nm+1 leaves and nm-1 internal nodes. The time-complexity of propagation by *TreeProp-N* can be estimated as $i(nm+1+3(nm-1))=O(inm)$, because each internal node of a binary tree has three neighbors. In the case of TreeProp-E the computation is similar, except that the internal edges have four neighbours, but the overall time complexity remains $O(inm)$. We point out that i) the size of the tree is much less than nm since - owing to the clustered nature of the protein similarity space - the lists of the nearest neighbors are largely overlapping; ii) the number of iterations is typically less than *20*, and iii) applied in this way, the algorithms will reorder just the top ranking elements of the original list of similarities. *TreeProp-N* and *TreeProp*-E were written in MatLab using the Bioinformatics Toolbox. For the timing of RankProp we ported the original C source code of J. Weston into MatLab. Table **1** gives a summary of the approximate time requirements for each algorithm.

Table 1. Wall Clock Time Requirements for the *RankProp, TreeProp-N* and *TreeProp-E* algorithms[1]

	***RankProp*[2]**	***TreeProp-N*[3]**	***TreeProp-E*[3]**
BLAST search[4]	*1721.39*	*1721.39*	*1721.39*
Tree-building	–	*1180.13*	*1180.13*
Propagation	*15528.2*	*204.14*	*205.46*
Total:	*17249.59*	*3105.66*	*3106.98*

Table **1** shows that both tree-based algorithms faster than network-based propagation, and the gain in time compensates for the extra time-requirement of tree-building. Naturally, network-based propagation (*RankProp*) runs faster if applied to a smaller sized network, but this results in a decrease in performance as mentioned in Section 6.

In practical tests, the parameters of RankProp were the same as those recommended by Weston *et al.* in [8], i.e. the α parameter was *0.95* and the number of iteration was set to *20*. In the case of tree-based methods we used $\alpha = 0.3$ and $i=20$. These values were chosen because i) changing α parameter between *0-0.5* resulted in little variation in performance, and ii) *TreeProp-N* an *TreeProp-E* were typically found to converge in *10* steps or fewer.

The dependence of the performance on alpha and the number of iterations is shown for *TreeProp-N* in (Fig. **2**). The dependence is roughly similar for *TreeProp-E* (not shown).

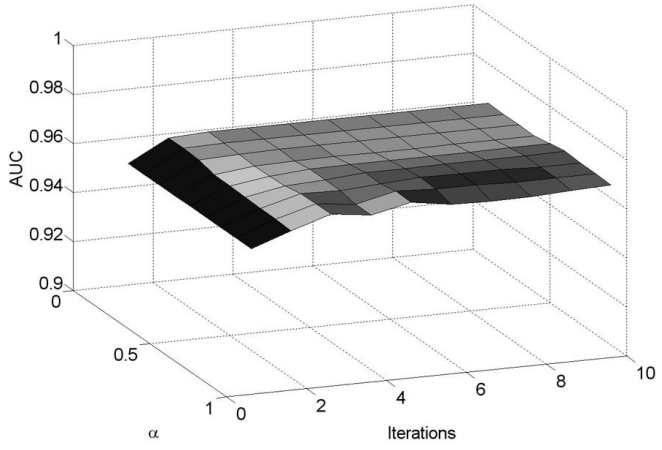

Figure 2. Ranking performance of TreeProp-N as a function of the alpha parameter in Eq. 4 and the number of iteration steps. The ranking performance is the cumulative ROC AUC value (Section) calculated on the 3PGK dataset.

5.2. Performance Evaluation on Various Databases

The algorithms were evaluated in terms of ROC analysis in the way described in Section 3.1. We also calculated the AUC_{50} (ROC_{50}) value [20], because these values are not so sensitive for the class imbalance caused by a large excess of negatives compared to positives, which is a typical situation with all protein datasets analyzed below. In addition to the propagation algorithms (*RankProp, TreeProp-N, TreeProp-E*) we used the simple nearest neighbor evaluation (1NN) as a basis of comparison.

We tested the methods on three protein datasets (3PGK, SCOP40mini, COG) using sequence similarity (BLAST, Smith-Waterman) and/or structural similarity (DALI) [21]. The datasets were selected so as to represent various degrees of difficulty. In the 3PGK dataset, the similarity between group members is high, and between various groups it is also quite high. In SCOP40mini, both the within-group and the between-group sequence similarities are relatively low. In COG, the within-group similarities are high and the between-group similarities are low. Since there are several ten to several hundred classification tasks defined on each datasets, we used the cumulative AUC value as a performance indicator. In a few cases (Tables **4-5**) the cumulative *AUC* was close to *1.00*. In these cases we selected a few problematic tasks for the comparison. Tables **2-3** show the results obtained for sequence comparison methods BLAST and Smith-Waterman. In general, the performance of *TreeProp-N* and *TreeProp-E* are similar to each other and slightly surpass that of *RankProp* followed by 1NN. Out of the 136 cases (for both similarity measure, SW, BLAST), *RankProp* and 1NN were the winners in 28 and 34 cases, respectively.

Table **5** contains data on structural comparison on the SCOP40mini dataset. This dataset is complex if we use sequence comparison (Table **3**), but it is relatively easy if we use an efficient 3D comparison such as DALI. In this case the cumulative AUC was so high that there was hardly any difference between the algorithms, so we chose a few of the most problematic cases for comparison. Even though the AUC values are high, we see the same pattern as we do with sequence comparisons, i.e. in general there is an improvement caused by propagation, and *TreeProp-N* and *TreeProp-E* both perform well compared to *RankProp*.

6. DISCUSSION AND CONCLUSIONS

Phylogenetic trees are used by biologists to highlight the salient internal structure of the protein universe in the form of a "tree of life". The rationale behind using phylogenetic trees for propagation is based on two assumptions; *i)* Propagation on the salient edges of network may increase the efficiency of the process due to noise reduction; *ii)* tree structures are sparse compared to a full network, so the process will be faster. Here we employed two strategies. *TreeProp-N* follows the strategy of *PageRank*, the only difference being that the propagation is applied to a tree, rather than to an entire network. *TreeProp-E* on the other hand propagates the edge-weights to neighboring edges. A further difference with

Table 2. Comparison of the Algorithms on the 3pgk Dataset Using Smith-Waterman and BLAST Scores[1]

	Smith-Waterman		BLAST	
3pgk	AUC	AUC_{50}	AUC	AUC_{50}
1NN	0.892	0.892	0.899	0.899
RankProp[2]	0.961	0.961	0.963	0.963
TreeProp-N[2]	0.954	0.954	0.951	0.951
TreeProp-E[2]	**0.967**	**0.967**	**0.964**	**0.964**

Table 3. The AUC Values on the SCOP40mini Dataset Using Smith-Waterman and BLAST Scores[1]

	Smith-Waterman		BLAST	
SCOP40mini	AUC	AUC_{50}	AUC	AUC_{50}
1NN	0.815	0.781	0.763	0.774
RankProp[2]	**0.88**	0.76	0.725	0.655
TreeProp-N[3]	0.86	**0.797**	0.792	**0.808**
TreeProp-E[4]	0.859	0.678	**0.799**	0.754

[1]The raw scores were used for the comparison. [2]The propagation was carried out in the same way as before. [3]The propagation was carried out for the n = 20 top-ranking entries and there m = 20 neighbors, in i = 20 steps.

Table 4. Comparison of the Algorithms on Selected Classification Tasks Defined on the COG Dataset Using BLAST Scores[1]

COG	Eval.	Classification tasks										
		COG0631	COG0695	COG0697	COG0699	COG0814	COG0842	COG0847	COG1310	COG1752	COG2036	Mean
1-NN	AUC	0.899	0.953	0.985	0.905	0.923	0.952	0.83	0.698	0.948	0.999	0.924
	AUC_{50}	0.977	**1**	0.864	**1**	0.885	0.927	0.98	1	0.991	0.996	0.969
RankProp[2]	AUC	0.862	**0.978**	0.97	0.818	0.912	0.908	0.797	0.47	0.967	1	0.877
	AUC_{50}	0.975	0.978	0.662	0.922	**0.947**	0.693	0.933	**1**	0.898	1	0.829
TreeProp-N[3]	AUC	0.974	0.95	0.998	0.96	0.91	0.94	0.949	0.978	0.933	1	**0.966**
	AUC_{50}	0.96	0.98	**0.947**	0.996	0.945	**0.977**	**0.997**	0.889	0.98	1	**0.969**
TreeProp-E[3]	AUC	**0.976**	0.949	**0.998**	0.886	0.913	0.945	0.954	0.689	0.938	1	0.941
	AUC_{50}	**1**	0.944	0.929	0.996	0.945	**0.977**	**0.997**	0.889	0.98	1	0.969

[1]The raw scores were used for the comparison. [2]The propagation was carried out in the same way as before. [3]The propagation was carried out for the n = 20 top-ranking entries and there m = 20 neighbors, in i = 20 steps. For the evaluation of the results see text.

respect to *PageRank* is the fact that the ranking is carried out according to the distance within a binary tree, rather than by using simple a similarity/distance measure.

Using ROC analysis on protein datasets (a total of *84* sequence and structure classification tasks) of varying difficulty, we found that *TreeProp-N* and *TreeProp-E* perform somewhat better and are faster than the *personalized PageRank/RankProp* algorithm.

In the practical implementation of *TreeProp-N* and *TreeProp-E* we used the FastME algorithm [14] for tree construction. We also tested BioNJ [22] and Weighbor [23], which provided identical results on our datasets, but required more cpu time (data not shown).

Table 5. Comparison of the Algoritnms on Selected Classification Tasks Defined on SCOP40mini Dataset, Using the DALI 3D-Comparison Scores[1]

SCOP40mini	Evaluation	Classification tasks								
		a.39.1._a.39.1.5.	a.4.1._a.4.1.1.	b.1.18._b.1.18.2.	b.1.1._b.1.1.3.	c.2.1._c.2.1.3.	c.37.1._c.37.1.9.	c.47.1._c.47.1.1.	d.81.1._d.81.1.3.	Mean
1-NN	AUC	0.949	0.967	0.988	0.96	0.961	0.903	0.99	0.948	0.958
	AUC_{50}	0.985	0.567	0.857	0.706	0.966	0.735	0.936	0.989	0.843
RankProp[2]	AUC	0.905	0.976	0.978	0.969	0.824	0.961	0.999	0.971	0.948
	AUC_{50}	1	0.685	0.723	0.735	0.98	0.542	0.991	1	0.832
TreeProp-N[3]	AUC	0.946	1	0.996	0.968	0.997	1	1	0.993	0.987
	AUC_{50}	1	1	0.96	0.883	0.964	1	1	1	**0.976**
TreeProp-E[3]	AUC	0.949	1	0.998	0.987	0.993	1	1	0.996	**0.99**
	AUC_{50}	1	1	0.917	0.851	0.976	1	1	1	0.968

[1]The raw scores were used for the comparison. [2]The propagation was carried out in the same way as before. [3]The propagation was carried out for the n = 20 top-ranking entries and there m = 20 neighbors, in i = 20 steps.

For both tree-based algorithms we used a reduced network, namely we selected the *n* top-ranking objects (*n=40*) and their *m=40* nearest neighbors. Increasing *n* and *m* beyond this value did not affect the performance, and values of *m*=20 were generally satisfactory. In principle, this size reduction will make the computation faster, but it can also act as a noise reduction filter, since we select only the "important neighbors" of the query. The question arises whether or not the same toplist restriction strategy would improve the performance of *PageRank/RankProp*. However when we used a reduced network (n=m=40) in conjunction with RankProp on the SCOP40 mini dataset (*1337* proteins, *55* classification tasks), the AUC value dropped from *0.880* to *0.756*. We obtained similar results on the other datasets as well (data not shown) so we believe that the improvements in performance were not induced by the toplist restriction being applied.

Summarizing, we can conclude that tree-structures can be efficiently used to increase the performance of protein classification algorithms. Even though we tested our algorithms on a large variety of classification test, we think that the improvements may critically depend on the applications, thus the parameters may need to be adjusted for the datasets that will be analyzed.

APPENDIX: DATASETS AND METHODS OF CALCULATION

Phylogenetic Tree Building Methods

Biologists use a whole arsenal of sophisticated methods for building binary phylogenetic trees. One of the most popular ones is the Neighbor-Joining [24] and its more recent variants BioNJ [22] and Weighbor [23], which are known to produce consistent trees in $O(N^3)$ time provided we have additive distance. Since we needed to build up a tree for each query protein, we looked for fast and sensitive equivalents and chose the FastME algorithm that uses the Greedy Minimum Evolution tree construction method with Nearest Neighbor Interchange operator, and requires only $O(N^2)$ time [14]. The performance of FastME compares favorably with that of other, state-of-the-art algorithms [14]. We used the C implementation of FastME downloaded from http://www.ncbi.nlm.nih.gov/CBBresearch/Desper/FastME.html.

Performance Evaluation

The evaluation was carried out via standard receiver operator characteristic (ROC) analysis [25]. This method is especially useful for protein classification as it includes both sensitivity and specificity, based on a ranking of the objects to be classified [20]. In the case of 1NN classification the ranking variable was the nearest neighbor similarity or distance value calculated between a query sequence on the one hand and the members of the positive training set on the other. This scenario corresponds to a one-class classification with outlier detection. In short, the analysis was carried out by plotting sensitivity vs. 1-specificity at various threshold values, then the resulting curve was integrated to give an "area under curve" or AUC value. If the evaluation procedure contained several ROC experiments, one can draw a cumulative distribution curve of the AUC values [20], or use the average of the individual AUC values can be used as performance indicator [26,27].

Datasets

The protein datasets were taken from the Protein Classification Benchmark Collection (PCBC, [28]). The 3PGK dataset contains *131* proteins of identical function (id: PCB00016), divided into 10 classification tasks. The SOP40mini dataset contained *1357* proteins grouped by 3D

structure (id: PCB00019), divided into *55* classification tasks. The COG dataset contained *17973* proteins grouped by function (id: PCB00017), divided into *117* classification tasks. From this dataset we evaluated only a few "difficult" tasks in order to test our algorithm.

7. ACKNOWLEDGMENTS

A. Kocsor was supported by the János Bolyai fellowship of the Hungarian Academy of Sciences. Work at ICGEB was supported in part by grants from the Ministero dell.Universita‘ e della Ricerca (D.D. 2187, FIRB 2003 (art. 8), "Laboratorio Internazionale di Bioinformatica"). We are grateful to an anonymous reviewer for calling our attention to references [6] and [7].

LIST OF ABBREVIATIONS

ROC = Receiver Operator Characteristic

AUC = Area Under Curve

PCBC = Protein Classification Benchmark Collection

NJ = Neighbor joining

1-NN = Nearest Neighbor Classifier

REFERENCES

[1] Lauritzen, S. L.; Spiegelhalter, D. J. **(1988)** *Journal of the Royal Statistical Society*, *50*, 157-224.

[2] Pearl, J.**(1988)** Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference; Morgan Kaufmann.

[3] Brin, S.; Page, L. (**1998)** *Computer Networks and ISDN Systems*, *30*, 107-117.

[4] Motwani, R.; Raghavan, P. **(1995)** Randomized Algorithms (Cambridge International Series on Parallel Computation); Cambridge University Press.

[5] Liao, L. and Noble W. S. **(2003)** *J. Comput. Biol., 10*, 857-868.

[6] Zhao X.M.,Cheung Y.M. and Huang D.S. **(2005)** *Neural Networks,18,* 1019-1028

[7] Huang D.S., Zhao X.M., Huang G.B., and Cheung Y.M. **(2006)** *Pattern Recognition, 39,* 2293–2300

[8] Weston, J.; Elisseeff, A.; Zhou, D.; Leslie, C. S.; Noble, W. S. **(2004)** *Proc. Natl. Acad. Sci. U S A*, *101*, 6559-6563.

[9] Kuang, R.; Weston, J.; Noble, W. S.; Leslie, C. (**2005)** *Bioinformatics*, *21*, 3711-3718..

[10] Leone, M.; Pagnani, A. (**2005)** *Bioinformatics*, *21*, 239.

[11] Zhou, D.; Weston, J.; Gretton, A.; Bousquet, O.; Schölkopf, (**2004),** *Advances in Neural Information Processing Systems, 16*, 169-176..

[12] Hegyi, H.; Pongor, S. (**1993)** *Bioinformatics(CABIOS)*, *3*, 371-372.

[13] Murvai, J.;Vlahovicek, K.; Barta, E.; Parthasaraty, S.; Hegyi, H.;Pfeiffer, F.;Pongor, S. (**1999)** *Bioinformatics*, *4*, 343-344.

[14] Desper, R.; Gascuel, O. **(2002)** *J. of Comp. Biol.*, *5*, 687-705.

[15] Hassan S. and Banea C. **(2006)** In Workshop on TextGraphs, at HLT-NAACL, 53–60.

[16] Mihalcea, R.; Tarau, P.; Figa, E. **(2004)** In *Proc. of 20st Int. Conf. on Comp. Ling.*, Article No. 1126.

[17] Parlet, B. N. **(1994)** *The symmetric eigenvalue problem*; Prentice-Hall,Englewood, Cliffs, NJ.

[18] Smith, T. F.; Waterman, M. S. **(1981)** *J. Mol. Biol.*, *147*, 195-197.

[19] Altschul, S. F.; Gish, W.; Miller, W.; Myers, E. W.; Lipman, D. J. **(1990)** *J. Mol. Biol*, *215*, 403-410.

[20] Gribskov, M.; Robinson, N. **(1996)** *Comput. Chem.*, *20*, 25-33.

[21] Holm, L.; Sander, C. **(1995)** *Trends Biochem Sci.*, *11*, 478-80.

[22] Gascuel, O. **(1997)** *Mol. Biol. Evol.*, *14*, 685-695.

[23] William, J. B.; Nicholas, D. S.; Aaron, L. H. **(2000)** *Mol. Biol. Evol.*, *1*, 189-197.

[24] Saitou, N.; Nei, M. **(1987)** *Mol. Biol. Evol.*, *4*, 406-425.

[25] Egan, J. P. **(1975)** *Signal Detection theory and ROC Analysis.*; New York: Academic Press.

[26] Sonego P., Kocsor A. and Pongor S. **(2007)**, *Briefings in Bioinformatics*, in press.

[27] Busa-Fekete R, Kertész-Farkas A, Kocsor A, Pongor S. (2007), *J Biochem. Biophys. Methods.,* doi:10.1016/j.jbbm.2007.06.003

[28] Sonego, P.; Pacurar, M.; Dhir, S.; Kertész-Farkas, A.; Kocsor, A.; Gáspari, Z.; Leunissen, J. A. M.; Pongor, S. **(2007)** *Nucleic Acids Res.*, *35*, 232-236.

Received: ?????????????? **Revised:** ?????????????? **Accepted:** ??????????????