# An Approximate de Bruijn Graph Approach to Multiple Local Alignment and Motif Discovery in Protein Sequences

Rupali Patwardhan[1], Haixu Tang[1,2], Sun Kim[1,2], and Mehmet Dalkilic[1,2]

[1] Center for Genomics and Bioinformatics, Indiana University, 1001 E. 3rd Street, Bloomington, IN 47405
[2] School of Informatics, Indiana University, 901 E. 10th Street, Bloomington, IN 47408
{rpatward, hatang, sunkim2, dalkilic}@indiana.edu

**Abstract.** Motif discovery is an important problem in protein sequence analysis. Computationally, it can be viewed as an application of the more general multiple local alignment problem, which often encounters the difficulty of computer time when aligning many sequences. We introduce a new algorithm for multiple local alignment for protein sequences, based on the de Bruijn graph approach first proposed by Zhang and Waterman for aligning DNA sequence. We generalize their approach to aligning protein sequences by building an approximate de Bruijn graph to allow gluing similar but not identical amino acids. We implement this algorithm and test it on motif discovery of 100 sets of protein sequences. The results show that our method achieved comparable results as other popular motif discovery programs, while offering advantages in terms of speed.

**Keywords:** Motif discovery, local alignment, de Bruijn graph, proteins

## 1 Introduction

As an important problem in bioinformatics, motif discovery in protein sequences can be viewed as a direct application of a more general multiple local alignment problem. Programs like MEME [1], Gibbs Sampler [2] and BlockMaker[3] represent popular solutions to this problem and have been commonly used in many applications. However, these programs share the same shortcoming of the computational efficiency, preventing their applications to large scale cases.

Zhang and Waterman recently introduced a fundamentally novel algorithm for local multiple alignment of DNA sequences [4]. This was an extension of their EulerAlign approach for global multiple alignment [5]. They aim to construct a consensus pattern that is most consistent with all input sequences from a de Bruijn graph built from the input sequences. The consensus is then used as a query to locate all instances of the pattern. Their algorithm has the obvious advantage over the previous methods that it reduces the time complexity of the problem to approximately linear, thus significantly reducing the computational time when the input size is large.

In spite of its success with DNA sequences, it is not straightforward to apply this method to local multiple alignment of protein sequences. There are several important distinctions between homologue protein and DNA sequences. First, protein sequences use a larger alphabet (with 20 amino acids) than DNA sequences (with 4 nucleotides). Second, homologous protein sequences have lower sequence identity than DNA sequences. For instance, even closely homologous protein sequences may have only 50% $\sim$ 60% identity; let alone the fact that some distantly homologous protein sequences may have as low as 30% identity. As a result, the possiblity of finding identical $k$-mers that would form the basis of the consensus path is much lower. Finally, amino acid residues, represented by 20 letters in protein sequences, are not equally similar with each other. [3] For example, leucine residue is more similar to other hydrophobic amino acids, e.g. isoleucine or valine, than the hydrophilic residues. These differences are represented in amino acid similarity matrices, such as PAM [6] or BLOSUM [7].

The above three differences complicate the application of de Bruijn graph approach to the protein sequence alignment. In this paper, we attempt to address these issues by introducing the concept of *approximate de Bruijn graph*. Then we adopt a similar approach as Zhang and Waterman to perform the local multiple sequence alignment by traversing this graph. We first find a heaviest path in the approximate de Bruijn graph using a heuristic greedy algorithm and deduce a consensus protein sequence from this path. Next we align this consensus sequence to each of the input sequence and finally construct the multiple alignment from these pairwise alignments. We then repeat this to find other patterns.

We applied this method to motif discovery in protein sequences. The algorithms were tested on protein families corresponding to PROSITE [8] patterns, and also compared against two other existing motif discovery algorithms, MEME [1] and PRATT [9]. Results show our algorithm outperforms PRATT and generates comparable results to MEME, but runs much faster. We also present a modified version of this algorithm, which attempts to account for motifs that contain certain non-specific positions or gaps. In case of motifs with no continuous conserved amino acid residues but with at least one occurrence of alternate conserved residues, our algorithm outperformed both MEME and PRATT.

## 2 Approximate de Bruijn Graph

Consider $n$ protein sequences as input. A $k$-tuple is a subsequence of length $k$ in one of the sequences. In de Bruijn graph [10], we represent each $k$-tuple by two connected nodes, in which each node represents a $k-1$-tuple and these two $k-1$-tuples overlap with $k-2$ letters in the original sequence; the connecting edge represents the $k$-tuple pointing from the prefix $k-1$ tuple to the suffix $k-1$

[3] Letters in DNA sequences, i.e. nucleotides, are not equally similar with each other either, as the mutation rate of transition, i.e. the mutation from purine to purine, or from pyrimidine to pyrimidine, and transversion, i.e. the mutation from purine to pyrimidine or *vice versa* differs. But the difference is subtle and ignored in DNA sequence alignment.

tuple (Fig. 1 (b)). Two identical $k$-tuples in the input sequences correspond to the same edge in the de Bruijn graph. The *weight* of each edge is defined as the number of identical $k$-tuples in the input sequences. The de Bruijn graph of the input sequences can be constructed in a progressive way. First we process one input sequence at a time and add the $k$-tuples in this sequence into the graph. If the incoming $k$-tuple corresponds to an edge existing in the current graph, we increase the weight of this edge by 1; otherwise we create a new edge (and nodes if necessary) accordingly and assign weight 1 to this edge. It is easy to show that this construction procedure is independent of the order in which input sequences are processed.

In order to take into account the similarity between amino acid residues, we define the *approximate de Bruijn graph*, which generalizes the classical definition of de Bruijn graph. Given $n$ protein sequences, *approximate* de Bruijn graph has the same topology (nodes and edges), but different edge weights as compared to the *classical* de Bruijn graph. If one $k$-tuple is *similar* to $w$ other $k$-tuples in the de Bruijn graph, the weight of the corresponding edge in approximate de Bruijn graph is incremented by the sum of weights of all these $w$ $k$-tuples scaled by the degree of similarity and a similarity constant. If we adopt the rigorous criteria in defining the *similarity* between two $k$-tuples, *approximate* de Bruijn graph is reduced to the *classical* de Bruijn graph. But generally we want to relax this rigorous condition to account for amino acid similarities. We define two $k$-tuples $a$ and $b$ are *similar*, if the substitution score for each pair of corresponding residues in the two subsequences represented by two edges is larger than a predefined threshold $\theta$. The substitution score between each pair of residues can be extracted from any standard protein similarity matrices such as BLOSUM or PAM or any other user defined similarity matrix.

## 3 Algorithm for Motif Discovery

Given $n$ protein sequences, our algorithm for motif finding involves four major steps:

1. Build the classical *de Bruijn graph* from the set of input sequences;
2. Transform the *classical* de Bruijn graph into *approximate* debruijn graph by adjusting the weights of edges in the graph taking into account the amino acid similarities;
3. Traverse the *approximate de Bruijn graph* to discover significant consensus subsequences;
4. Locate all instances of the reported subsequences identified in the previous step and output their multiple alignment;

In the next section we will elaborate on these steps.

### 3.1 Building the Classical de Bruijn Graph

Let each edge in the graph correspond to a subsequence of length $k$. We adopt the following progressive procedure to build the *classical* de Bruijn graph from

a set of given protein sequences.
We process the sequences one at a time, and for each sequence,

1. Move along the sequence shifting one amino acid position at a time;
2. At each position consider a subsequence of length $k$ starting at that position;
3. Check if the edge corresponding to this subsequence already exists in the graph;
4. If it does, then increment the weight of that edge by 1; otherwise, create a new edge and assign it a weight of 1.

### 3.2 Adjusting Edge Weights

We define two edges to be *similar* if every pair of corresponding residues in the two subsequences represented by two edges has a positive substitution score as per a similarity matrix. This similarity matrix can be a standard substitution scoring matrix such as BLOSUM62 (used as default in this study) or any other user defined similarity matrix.

Consider two edges X and Y. They will be considered to be *similar* if and only if

$$s(X_i, Y_i) > \theta, \ \forall_{i:1 \leq i \leq k} \tag{1}$$

Where,
$X_i = i^{th}$ amino acid residue in the $k$-tuple represented by the edge X,
$Y_i = i^{th}$ amino acid residue in the $k$-tuple subsequence represented by the edge Y,
$s(X_i, Y_i)$ = substitution score for residues $X_i$ and $Y_i$ as per the substitution matrix.
$\theta$ = residue similarity threshold 0 as default

The weights of similar edges are incremented proportional to their similarity. For this, we introduce a relative similarity score as a measure of similarity between edges. A *relative similarity score* $r$ between two edges X and Y is defined as,

$$r(X, Y) = \sum_{i=1}^{k} \frac{s(X_i, Y_i)}{s(X_i, X_i)} \tag{2}$$

To incorporate similarity into the graph, for each edge in the graph,

1. Check if any other edges are similar to it.
2. If yes, update the weight of this edge by adding to it the weights of these similar edges scaled by their similarity score and a user defined similarity constant (default 0.5);otherwise keep the weight unchanged.

For example, suppose edge A is under consideration. We find that edges B, C and D are similar to it. Then we update the weight of edge A as follows,

$$W'_A = W_A + K \times (r(A,B) \times W_B + r(A,C) \times W_C + r(A,D) \times W_D) \quad (3)$$

Where,
$W_X$ = weight of edge X in classical de Bruijn graph,
$W'_X$ = weight of edge X in approximate de Bruijn graph,
$K$ = similarity constant

At the end of this step, we construct the approximate de Bruijn graph. The topology of this graph is the same as that of the classical de Bruijn graph. Only the weights of some edges are altered. We note that as in case of a classical de Bruijn graph, the weights as per this scheme are independent of the order in which the edges are processed. The similarity constant $K$ can be used to adjust the effect of similar edges. Thus, if $K = 0$, the weights of edges in approximate de Bruijn graph is the same as in classical de Bruijn graph.

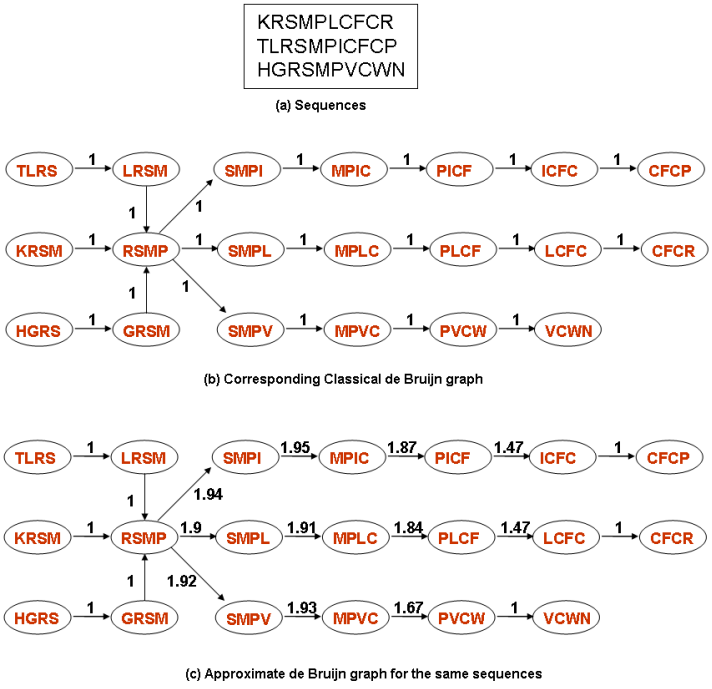


**Fig. 1.** *Classical* (b) and *approximate* (c) de Bruijn Graphs of a given set of protein sequences (a).

A small example shown in Fig.1 illustrates the difference in edge weights before and after similarity adjustment, and how the new weights help in identification of the motif. Figure 1(a) shows three input protein sequences. Figure 1(b) represents the classical de Bruijn graph built from these sequences. Figure 1(c) illustrates the graph after weights have been adjusted to make it *approximate*.

### 3.3 Traversing the Graph

A high weight for an edge in de Bruijn graph indicates that the subsequence represented by the edge and its similar subsequences occur many times in the input sequences. Therefore, we attempt to identify the highly weighted (*heavy*) paths by traversing the graph:

1. Identify the heaviest edge. The $k$-tuple represented by this edge is used as a seed of the consensus.
2. Starting from this seed, successive edges in the graph are traversed. At each step, the seed is extended to include the edge being traversed. Motif extension continues until the weight of the traversed edge is below a threshold value i.e. a percentage (defined by the user, default 80%) of the maximum weight of all edges in the initial approximate de Bruijn graph. The same procedure is applied to both sides of the seed. Each time there is more than one edge to traverse, we choose the one with higher weight. We then extract a consensus motif from the final path.
3. Using this consensus, identify all the occurrences of the consensus motif in the input sequences;
4. Decrease the weights of edges in the identified path [4];
5. The process is repeated until all seed-containing paths with a weight above the threshold are identified. Again this threshold is a percentage of the global maximum weight edge, but may be different from the threshold for seed extension used above.

### 3.4 Building Multiple Local Alignment

Once all the consensus sequences are identified, we use a linear space implementation of the Smith-Waterman local pairwise alignment [11],[12] available as a part of the FASTA package to align the consensus with each input sequence. The resulting pairwise alignments are then coverted to a local multiple alignment.

## 4 Algorithm for Gapped Motif Discovery

The above algorithm is very effective when the anticipated motif contains several continuous conserved residues that correspond to a highly weighted path

[4] This is similar to the declumping procedure described by Zhang and Waterman [4]. In our case, weights of similar edges should also be appropriately adjusted

in the approximate de Bruijn graph. However, the method may fail if the motif consists of alternating conserved and non-conserved residues, e.g. AxAxCxDx-AxGxC ('x' represents a non-specific position and may be any kind of residue) or AxCDxGxRGxC. As there is no restriction on 'x's, residues in this position in different motif instances could have negative substitution scores with each other, thus may not be over the residue similarity threshold $\theta$. In such cases, each occurrence of the motif leads to separate non-similar edges in the graph, and hence their total weight is not high enough to be detected using a reasonable similarity threshold. Lowering $\theta$ obviously is not a good option because it will then increase false positive identifications. To address this issue, we came up with a slightly modified *gapped* version of the algorithm. The only difference between this new algorithm and the algorithm described above is in the first step of building the graph. In the new algorithm, we create two types of nodes, in addition to the nodes for the $k$-tuples in the input sequences, we also create nodes for *masked* (by *non-specific masks*) tuples.

### 4.1 Masking Subsequences

Let '1' represent a conserved amino acid and '0' represent a gap or *non-specific position*. A *mask* can then be defined as a string of 1s and 0s of length $k-1$. All permutations and combinations of 1's and 0's will yield $2^{k-1}$ such masks. However, masks with more zeros than ones will result in nodes that are too non-specific. So we only consider masks with number of 1's equal to or greater than the number of 0's. The masking operation can be considered analogous to a logical bit-wise AND. The residue that overlaps with 1 is retained while the residue corresponding to 0 is replaced by a 'x'. For example, if we apply the masking operation (denoted as '*') to the subsequence, ANCD, then ANCD * 1001 = AxxD, ANCD * 1101 = ANxD, ANCD * 1011 = AxCD and so on. Each subsequence in the input sequences will be masked using all pre-designed masks, and each resulting masked subsequence will be represented by a separate node in the graph (Figure 3).

### 4.2 Linking Edges

Edges are linked between nodes with overlapping subsequences as before. However the overlapping is defined more flexibly because of the 'x's, leading to a much more connected graph. As an example, consider 2 nodes of subsequences ANCD and RCDE. These nodes will not be linked in the approximate de Bruijn graph. But after masking, they will be linked by an edge. This is because, after masking ANCD with 1011, we get AxCD and this now can be connected to xCDE which is obtained by masking RCDE with 0111. Suppose we have three input sequences with subsequences as shown in Figure 2(a), they clearly contain a motif AxCDx. Accordingly, instead of three different edges each with a low weight (Figure 2(b)), one collapsed edge with a higher weight will be left (Figure 2(c)) after masking.

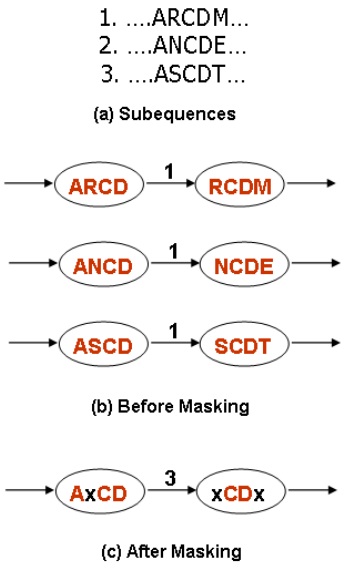


**Fig. 2.** Edge consolidation by masking subsequences in approximate de Bruijn graph. Three edges linked before masking (b) between input subsequences (a) collapse into one single edge with higher weight after masking (c).

The masking procedure enables us to detect motifs that could not be detected using the regular algorithm described previously. This is illustrated in Fig.3. Assume AxCDxxGH is the motif. Each instance of the motif will contain a non-specific residue at the 'x' position. Before masking, they would lead to separate edges each with low weight in the graph. Masking allows these instances to collapse into same edges, resulting higher weights to be detected by graph traversing (highlighted in red).

## 5 Results

We implemented both algorithms described above in Perl and tested the program on the motif discovery from sequences of the same protein families. Our program can be accessed through a website
http://biokdd.informatics.indiana.edu/rpatward/debruijn/project.html.

### 5.1 Test Datasets

We used a benchmarking data set for which the optimal motif is known and manually verified. PROSITE is a well known database of motif patterns, and their related protein families [8]. The PROSITE patterns are hand curated and

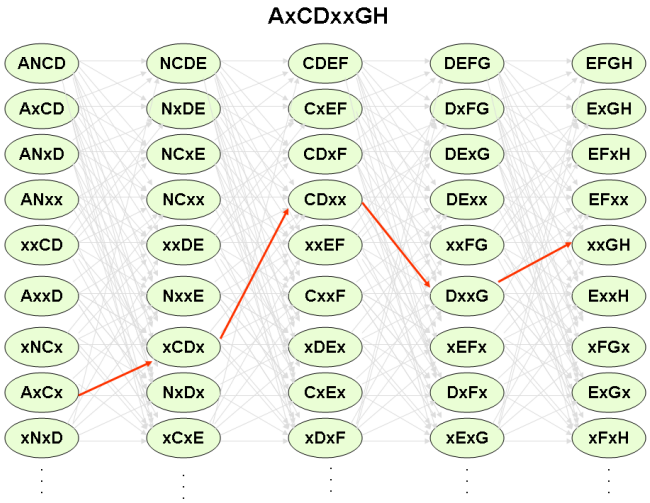


**Fig. 3.** An schematical example illustrating the advantage of masking procedure. After masking, motif instances (of AxCDxxGH) collapse into same edges, resulting a high weighted path that can be identified by graph traversing (highlighted in red).

correspond to sites that are proven to be biologically significant, and have already been used in the past by researchers to test the performance of motif discovery algorithms. In their paper, Hart et al [13] give a justification of the relevance of using overlap with PROSITE pattern sites as a good measure for estimating the quality and accuracy of a detected motif. Thus, we used protein families corresponding to PROSITE patterns as input, and the agreement with known PROSITE patterns as a measure of success or failure of the algorithm. We compared the performance of our algorithms with two other motif discovery algorithms, MEME and PRATT. We select these two programs since they represented two classes of conventional motif discovery methods: probabilistic (MEME) and combinatorial methods (PRATT). We note that it is likely that a motif output by either of the algorithms could be a true motif even if it is not the same one as the PROSITE pattern. But since it is not easy to ascertain this, we decided to consider just those that correspond to the respective PROSITE patterns as true motifs.

We used two different datasets. The first one consisted of a hundred protein families corresponding to first 100 PROSITE patterns in the database The second dataset (referred to as "hard-to-find" motif sets) included families corresponding to PROSITE patterns that had no continuous conserved amino acid residues, but at least one occurrence of alternating conserved residues.

As mentioned above, we first tested all programs on the first data set, i.e. protein families corresponding to first 100 PROSITE patterns that had a Data Bank Reference (PS00010 to PS00119). The families that were not included were those corresponding to 10 PROSITE entries PS00013, PS00015, PS00016,

PS00017, PS00029, PS00038, PS00040, PS00043, PS00044 and PS00107. Entries PS00013 and PS00015 are rules, not patterns. PS00016, PS00017 and PS00029 are categorized as too non-specific. PS00038, PS00040, PS00043 and PS00044 do not have a PROSITE entry associated with them at present. PS00107 was excluded because it is very non-specific; there more than one thousand sequences matching this pattern.

All four programs were run with default parameters. For MEME and PRATT, since we need to specify the number of motifs, the top three motifs were considered. It can be seen that our algorithms achieve comparable performance as the other popular algorithms. As we will show below, our program, however, runs much faster.

**Table 1.** Results for Test Dataset I

| Algorithm | No. of families |
|---|---|
| Approximate de Bruijn | 69 |
| Gapped de Bruijn | 75 |
| MEME | 80 |
| PRATT | 61 |

On the second data set, our gapped de Bruijn algorithm performed better than both MEME and PRATT, with the results agreeing with the PROSITE pattern for 128 (79%) of the 162 families (Table 2).

**Table 2.** Results for Test Dataset II

| Rank of motif | Gapped de Bruijn | MEME | PRATT |
|---|---|---|---|
| 1 | 68 | 50 | 77 |
| 2 | 26 | 26 | 13 |
| 3 | 12 | 10 | 4 |
| 4 | 12 | 6 | 6 |
| 5 | 10 | 10 | 5 |
| Total | 128 | 102 | 105 |

We note that both of our methods have their own advantages. The conventional method can detect similar segments even if they have no exact residues in common; however all the residues in the core motif should have similarity scores greater than the threshold $\theta$. On the other hand, the second (masked) method requires at least a couple of exactly conserved residues though they need not be continuous.

As an example consider the protein family correspoding to PS00050. The PROSITE signature for this family is $[RK](2) - [AM] - [IVFYT] - [IV] -$

$[RKT]-L-[STANEQK]-x(7)-[LIVMFT]$. The conventional approximate de Bruijn method can detect this motif, while the gapped method version cannot. But for another PROSITE pattern, PS00118 (signature $C-C-\{P\}-x-H-\{LGY\}-x-C$), the approximate method fails to identify this motif because the motif does not have enough conserved residues to form high weight edges, whereas the gapped method can successfully identify it.

To compare the speed of four programs, we plotted the running times of all four algorithms for a range of input sizes (Figure 4). Note that we consider the time of MEME running in parallel mode using 8 processors on IBM SP cluster whereas those of the other programs running on a single processor computer, for it runs significantly slower than the other programs.

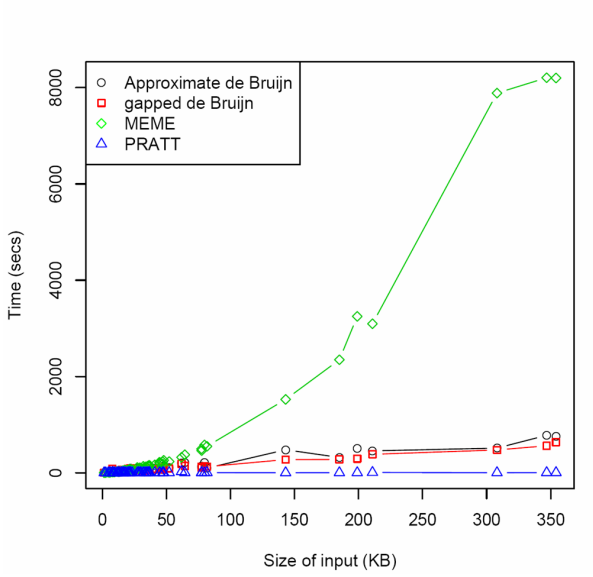


**Fig. 4.** Comparison of running times of four methods on data sets with different input size (X-axis).

## 6 Discussion

The results of our new methods look overall encouraging. It is competitive in terms of both speed as well as accuracy in comparison to existing algorithms. Furthermore, it scales very well for large inputs.

An elegant feature of the de Bruijn graph approach is that it is incremental. So in theory, even if you add one more input sequence to a precomputed data set, there is no need to start building the graph from scratch. It can be easily

added to the existing graph. Also, the graph structure and weights of edges and hence the result is independent of order of input sequences.

We used the BLOSUM or PAM substitution matrices to define the similarity of amino acid residues. In principle, any kind of user defined metric for similarity could be used for this purpose. This gives the users the ability to include their biological or intuitive knowledge in the motif discovery process.

When applied to motif discovery, our algorithm eliminates the need to specify the length and/or number of motifs in advance. All motifs that are above the specified motif threshold will be reported. However, unlike a lot of enumeration algorithms, this number is usually low so that the user does not have to sift through a long list of false positives to validate the real motifs.

In addition to motif discovery, a modification of our algorithm could be applied to protein classification, i.e. to determine the familyship of a protein sequence. This could be done by first building a de Bruijn graph from the sequences known to be in each protein family and then trying to traverse the graph using the query sequence. If the sequence belongs to this family, many of the nodes and edges required for traversing this sequence will already be present in the graph. On the other hand, if hardly any of the required edges exist in the graph, the sequence is apparently quite different from the sequences in the family and hence it is unlikely to belong to the family.

**Acknowledgements.** We thank members of the Bioinformatics Research Group at the School of Informatics, and the members of the Bioinformatics Team at Center for Genomics and Bioinformatics for many helpful discussions. Kim was partially supported by NSF CAREER DBI-0237901.

## References

1. Bailey, T.L., Elkan, C.: Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In: Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology, AAAI Press, Menlo Park, California (1994) 28–36
2. Lawrence, C., Altschul, S., Bogouski, M., Liu, J., Neuwald, A., Wooten, J.: Detecting subtle sequence signals: A gibbs sampling strategy for multiple alignment. Science **262** (1993) 208–214
3. Henikoff, S., Henikoff, J.G., Alford, W.J., Pietrokovski, S.: Automated construction and graphical presentation of protein blocks from unaligned sequences. Gene **163**(2) (1995) GC17–GC26
4. Zhang, Y., Waterman, M.S.: An Eulerian path approach to local multiple alignment for DNA sequences. PNAS **102**(5) (2005) 1285–1290
5. Zhang, Y., Waterman, M.S.: An eulerian path approach to global multiple alignment for dna sequences. Journal of Computational Biology **10**(6) (2003) 803–819
6. Dayhoff, M., Schwartz, R., Orcutt, B.: A model of evolutionary change in proteins. In: Atlas of Protein Sequence and Structure. Volume 5(3). National Biomedical Research Foundation (1978) 345–352
7. Henikoff, S., Henikoff, J.: Amino Acid Substitution Matrices from Protein Blocks. PNAS **89**(22) (1992) 10915–10919

8. Falquet, L., Pagni, M., Bucher, P., Hulo, N., Sigrist, C., Hofmann, K., Bairoch, A.: The prosite database, its status in 2002. Nucleic Acids Res. **30** (2002) 235–238
9. Jonassen, I.: Efficient discovery of conserved patterns using a pattern graph. CABIOS **13** (1997) 509–522
10. van Lint, J., Wilson, R.: A Course in Combinatorics. 2nd edn. Cambridge University Press (2001)
11. Myers, E.W., Miller, W.: Optimal alignments in linear space. CABIOS **4**(1) (1988) 11–17
12. Smith, T., Waterman, M.: Identification of common molecular subsequences. Journal of Molecular Biology **147** (1981) 195–197
13. Hart, R., Royyuru, A., Stolovitzky, G., Califano, A.: Systematic and fully automated identification of protein sequence patterns. Journal of Computational Biology **7(3-4)** (2000) 585–600