A Simple Algorithm for Detecting Circular Permutations in Proteins

S. Uliel[1], A. Fliess[1], A. Amir[2†], R. Unger[1*]

June 1999

[1]Faculty of Life Sciences
[2]Department of Computer Science and College of Computing, Georgia Tech
Bar-Ilan University
Ramat-Gan, 52900, Israel

[*] Corresponding author: ron@biocom1.ls.biu.ac.il
Tel: 972-3-5318124
Fax: 972-3-5351824

[†] Partially supported by NSF grant CCR-96-10170 and BSF grant 96-000509.

Abstract

Motivation: Circular permutation of a protein is a genetic operation in which part of the C-terminal of the protein is moved to its N-terminal. Recently, it has been shown that proteins that undergo engineered circular permutations generally maintain their three dimensional structure and biological function. This observation raises the possibility that circular permutation has been used by Nature during evolution. In this scenario a protein underwent circular permutation into another proteins, thereafter both proteins further diverged by standard genetic operations of insertions, deletions, and substitutions. To study this possibility one needs an efficient algorithm that for a given pair of proteins can detect the underlying event of circular permutations. A possible formal description of the question is: Given two sequences, find a circular permutation of one of them under which the edit distance between the proteins is minimal. A naive algorithm might take time proportional to N^3 or even N^4, which is prohibitively slow for a large scale survey. A sophisticated algorithm that runs in asymptotic time of N^2 was recently suggested, but it is not practical for a large scale survey.

Results: A simple and efficient algorithm that runs in time N^2 is presented. The algorithm is based on duplication of one of the two sequences, and then performing modified version of the standard dynamic programming algorithms. While the algorithm is not guaranteed to find the optimal results, we present data that indicate that in practice the algorithm performs very well.

Availability: A Fortran program that calculates the optimal edit distance under circular permutation is available upon request from the authors.

Contact: ron@biocom1.ls.biu.ac.il

1. INTRODUCTION

A circular permutation (CP) (throughout this paper we use the terms circular permutations and cyclic permutations interchangeably) of a sequence is a string operation in which a suffix of the sequence is "cut and pasted" to become a prefix of the sequence. Two proteins are related by a circular permutation when a fragment in the C terminal part of the first protein matches a fragment in the N terminal part in the second protein, and a fragment in the N terminal part of the first protein matches a fragment in the C terminal part of the second. See Figure 1 for a schematic view of a circular permutation of a protein sequence.

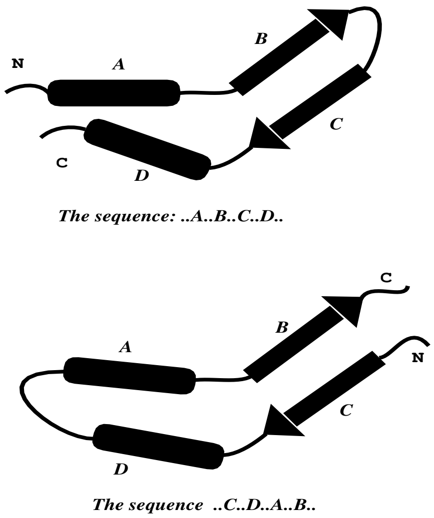

Figure 1: The original protein consist of four blocks: A,B,C,D. A circular permutation occur when the original N-terminal and C-terminal are joined, and new termini are formed by making a cut between the blocks B and C.

What are the structural and functional implications of circular permutations in proteins? Recently, several studies have been conducted to analyze this question. Protein were selected in which the N terminal of the protein are spatially close to the C terminal. Circular permutations were performed by joining, usually on the DNA level, the two termini and creating new termini by cutting the protein in another position along the sequence. The effects of these mutations were studied by comparing, using structural and functional assays, the properties of the mutated protein with the original. Few examples are (Goldenberg and Creighton, 1983; Luger *et. al.*, 1989; Ay *et. al.*, 1998; Hennecke *et. al.*, 1999), see also a review in (Heinemann and Hahn, 1995). In general, it was found that circular permuted proteins retain their three dimensional structure and biological function.

Since proteins that underwent circular permutation maintain their function, it raises the possibility that Nature has used circular permutations as constructive events in evolution. Indeed, there are few cases in which such permutation have been suggested to occurred naturally. I.e. pairs of natural proteins whose sequence relationship points to a circular permutation (Cunningham *et. al.*, 1979; Hemperly, 1983; Ponting and Russell, 1995; Heinemann and Hahn, 1995b; and a review by Lindqvist and Schneider, 1997). A possible evolutionary scenario would be that a gene was mutated by a genetic event of circular permutation to another protein, thereafter both proteins further diverge by the standard genetic events of insertions, deletions, and substitutions. The practical algorithm presented in this study is aimed to detect such a scenario from the sequence of two current proteins. In this work we formulate the question and explore possible solutions. A naive algorithm will require time complexity of N^3 or even N^4 in the length of the proteins. Recently, a theoretic algorithm was suggested (Landau *et. al.*,1998) that solve the problem in quadratic time. However, it is not practical for a large scale testing. We consider a practical approximation that can answer the question in time complexity of N^2 when N is the length of the proteins. We present data to show that the algorithm is efficient for actual biological sequences. We conclude by explaining how the algorithm could be used in a large scale survey to detect circular permutations in the protein database.

2. METHOD

Our long range goal is to search systematically for cases of circular permutations between pairs of proteins. It was indicated (Russell *et. al.*, 1998) that the standard tools of sequence comparison are not sensitive enough to identify many of the possible cases of CP. In some cases, sequence similarity tools can identify relatedness between a pair of proteins, but the fact that they are related specifically by circular permutation will be unnoticed. Furthermore, in cases where a protein had undergone a circular permutation early in its evolution, and both proteins accumulated a significant number of other genetic mutations later, it is reasonable that their alignment will not reveal any detectable similarity. But, if the original cyclic permutation can be identified and "corrected", it is conceivable that enough residual similarity exists and can be detected.

The most straightforward way to identify cyclic permutation is by looking for a characteristic pattern in a dot matrix format (Maizel and Lenk, 1981) of comparison between two sequences. Dot matrix (also known as dotplot) is a simple yet effective sequence comparison tool where human visualization is used to identify relationships between sequences such as similarity, repeats, and self-complementarity (Unger *et. al.*, 1986). The existence of two diagonal lines, off the main diagonal, which do not share rows or columns is a characteristic feature of clear-cut cases of CP. An example of such a matrix is shown in Figure 2(left) for the case of two lectin proteins: LEC_BOWMI and LECA_DIOGR (Swissprot names). The cyclic permutation between these two lectins is well known (Carrington et al., 1985), and occurs as a post translation event.

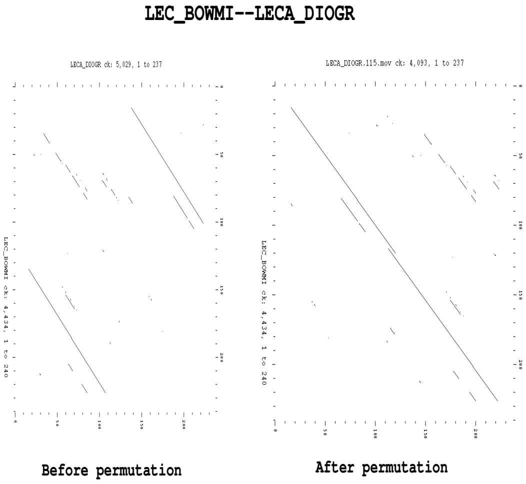

Figure 2: A dotplot comparison between two lectin proteins. When the proteins are compared "as-is" (left), two separate diagonal lines are seen. After applying the circular permutation found by the algorithm, the dotplot (right) shows a strong main diagonal.

When the appropriate circular permutation is applied (Figure 2,right), a main diagonal line is formed. Note however, that in more complicated cases of CP, where the homology between the two sequences have significantly diminished since the suspected CP event, or when the sequences have an internal repeat structure, visual identification is not always possible. It is also clear that this method is not suitable for large scale screening, since it requires human evaluation of each pairwise comparison.

Another alternative is using dictionary methods like Blast (Altschul *et. al.*, 1990) to search for circular permutations. These methods identify short fragments common to both sequences and report these best local matches. In principle this list of significant local matches can be analyzed to check for evidence for possible circular permutations. In practice, we found that it is difficult to automatically screen Blast output for CP. This is mainly due to the fact that Blast tend to break real alignments into small fragments and thus makes a unambiguous reconstruction problematic. Another problem is that in cases where the sequence similarity is low, then local hits whose specific combination could have been indicative for CP, might be below Blast detection level.

The most accurate and detailed method to search for relationship between two sequences is to calculate the full edit distance between them (global alignment). This algorithm, also known as Needelman and Wunsch (Needelman and Wunsch, 1970), measures the number of genetic operations of insertion, deletion and substitution needed to change one sequence into the other. Here we are interested in the following question: Given two sequences, find a circular permutation of one of them under which the edit distance is minimal. I.e. we are interested in finding out if there exist a CP event after which the two sequences can be significantly aligned.

This representation of the question lend itself to the following naive algorithm: Make all possible circular permutations of one protein relative to the other, and for each permutation calculate the regular edit distance by dynamic programming. The problem in this approach is its high computational cost. Each protein has N circular permutations. Thus, in theory there are N^2 different circular permutations between the two sequences. Note that because of the inherent asymmetry in the edit distance the edit distance $ED(A,B) \neq ED(A',B')$ where $A = (a_1, a_2, ...a_j, a_{j+1}, ...a_n), B = (b_1, b_2, ...b_j, b_{j+1}, ..., b_n), A' = (a_{j+1}...a_n, a_1, a_2, a_j), B' = (b_{j+1}, ..., b_n, b_1, b_2, b_j)$ i.e. when the same permutation was done on both sequences. However, it can be proven for the unit cost model (i.e. when all edit distance operation have a cost of one unit) that if all permutations of one of the strings, say A, are tested against string B, all edit distances of all *pairs of permutations* are achieved. Thus only N permutations are necessary and not N^2. In practice, the difference between $ED(A,B)$ and $ED(A',B')$ is dependent on the specific variant of the edit distance (especially on the form of the gap penalty) but is usually small, so only N permutations need to be tried. An edit distance calculation takes time proportional to N^2. Thus, the overall time requirement is N^3.

A time requirement of N^3 for a comparison between a pair of proteins is expensive. It becomes prohibitive when a full pairwise comparison of the database is needed. In our analysis a single N^2 comparison of sequence of size 300 amino-acids takes about 0.005 seconds (using a Silicon Graphics R10000 processor). An exhaustive search of all possible cyclic permutations for a single pair should take about 1.5 seconds (0.005*300). A complete survey of all pairs from the current protein database, which includes about 70,000 proteins, would take many CPU years(!).

An algorithm that can perform these comparisons in asymptotic time of N^2 rather than N^3 was recently suggested (Landau *et. al.*, 1998). Although this algorithm is asymptotically efficient and provides exact solutions, its implementation requires a number of doubly-linked and cross-linked data structures for the straightforward implementation. The better solution suggested there requires an additional use of suffix trees. These data structures and their manipulation multiply both the space and time necessary for the algorithm by a non-trivial constant (a conservative estimate would be around 30). Thus, a comparison of a pair of proteins of typical size would take at least (using the same processor as above) 0.15 second. A complete survey of the database of 70,000 proteins would thus require more than 7 CPU years (!). In addition, the algorithm in (Landau *et. al.*. 1998) specifically handles only unit-cost edit operations.

As an alternative, we present an efficient algorithm that, while it is not guaranteed to find the optimal permutation, does perform very well in practice. It is a variation of the edit distance algorithm in which one sequence is compared to a *duplication* of the other sequence, namely, for sequences A and B, A is compared to BB. The intuition here is that in this way, all consecutive permutations, each consist of a C-terminal fragment of B, followed by a N-terminal fragment of B, are compared to A, see figure 3.

Note however that many alignments that do not reflect a valid circular permutation become possible now, for example aligning sequence A with duplication of fragments from B (paying a penalty for the necessary deletions). The problem is how to enforce that the outcome of the algorithm will reflect the best alignment of the sequence A to a valid CP of B, rather than to a modification of B which is not a valid CP.

In order to improve its performance, we added two modifications to the classical edit distance algorithm. The first is initialization of the first row of the edit distance with zeros, instead of the usual initialization as $0, 1, 2, ..., N$. The standard sequential initialization for the first column is kept. The second modification that we made is to look for the value of the best alignment amongst all of the entries in the bottom row of the matrix. Actually,

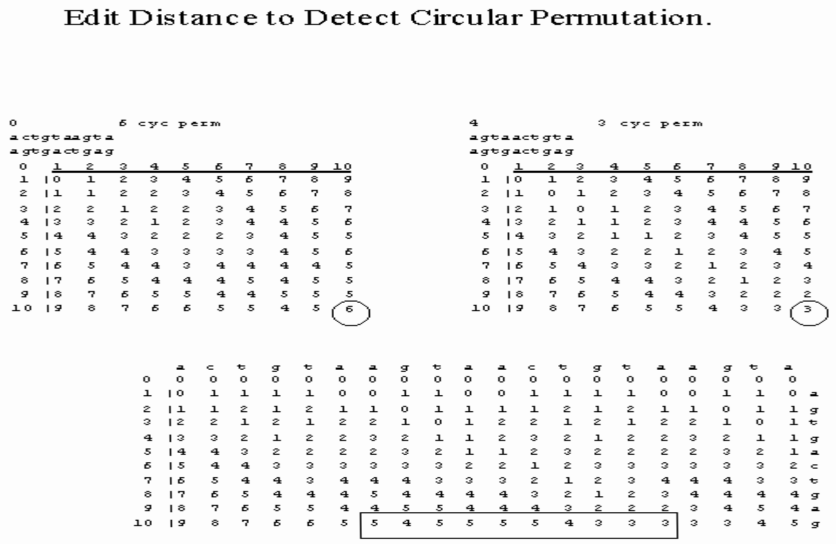

Figure 3: The two sequences on the top left are compared using dynamic programming and their edit distance is 6. All 10 circular permutations are tried, and for each the edit distance is calculated. The best cyclic permutation with the edit distance of 3 is shown on the top right. Bottom: the upper sequence is doubled and compared to the lower sequence. In a sense all permutations exist in this matrix, for example the marked part of the matrix corresponds to the permutation of the top right. The best result from all cyclic permutations can be retrieved as the lowest value in the last row of the matrix.

each entry in the right side of the last row, i.e. of columns $N+1, N+2, ., N+j, .., 2N$, can be seen as an approximation to the exact edit for a circular permutation of j characters respectively. These modifications are similar to the modifications of the standard edit distance algorithm to search for the best local alignment, known as the "Smith - Waterman" (Smith and Waterman, 1981) algorithm.

The valid optimal edit distance under CP is always a possible solution of the algorithm. Thus, every other solution that the algorithm produces must have lower value than the 'correct' value. In other words, the exact value for the edit distance for each permutation is an upper limit for the approximation.

The time and space complexity of the algorithm is clearly $2 \times N^2$ since the size of the dynamic programming matrix has been doubled on one dimension. With the same processor as above, a full survey of all pairs of proteins from the protein database will take about 8 CPU months. This time, especially when few workstations can share the computation load, make a complete survey feasible. As we mention in the discussion below, additional practical consideration can be used to further reduce the actual time needed.

3. RESULTS

When two sequences are related by circular permutation, it is expected that the edit distance after performing the "correct" circular permutation would yield a value that is much lower than the value of the edit distance directly comparing the two sequences, or the values resulting from "wrong" permutations. Since the last row of the edit distance matrix is used to approximate all the possible CPs, then a significant minimum along the sequence of values suggests a possible significant CP. As mentioned above, the values in this matrix are approximations, and should be verified by an exact calculation which consist of N calculations of edit distance costing N^2 each, one for every possible CP. An example of comparison of the approximation and the exact calculation for the case of lectin proteins for which the existence of circular permutation is well known (Hemperly and Cunningham, 1983), is given in Figure 4. In this example it is evident that the minima of both calculations coincide, giving validity to the approximation that is much quicker (about 120 times in this case) to calculate.

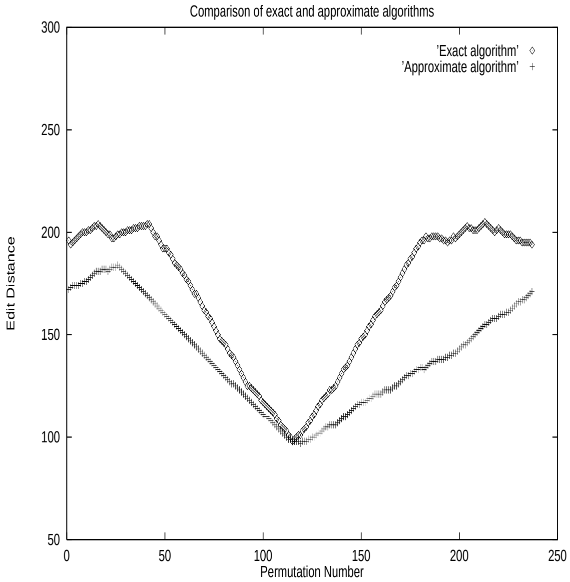

Figure 4: Comparison between the exact algorithm (i.e. calculating the edit distance for each circular permutation) and the approximate algorithm (as read from the last row of the doubled matrix) for two lectin proteins. The approximate algorithm follows the exact solution closely as a lower limit, and locates precisely the same minima.

In order to check the algorithm in a realistic situation we have tested the algorithm in the following way. For a given protein, we performed a random circular permutation followed by various levels of regular mutations (insertions, deletions, substitutions). So, for example we have pairs of proteins that when the circular permutation is corrected will show 50% of sequence similarity. The question that we are testing is to what extent the algorithm would be able to retrieve the underlying circular permutation. Two parameters were monitored, the percentage of cases in which the algorithm was able to retrieve the exact permutation, and the average distance of the found permutation to the 'correct' permutation. The results are shown in figure 5, for the protein erythroid 4.1 (411_CHICK in Swissprot) of length 90 residues. 100 examples were produced in each similarity level. The results show that when only a CP was performed, its retrieval rate is 100%. This is clear since when the edit distance under circular permutation is 0 (i.e. exact match under CP), our algorithm which calculates a lower limit, must also give 0. Even when two proteins have diverged significantly since the permutation, the algorithm was able to retrieve the circular permutation events. For example, for proteins that differ by 70% - 80%, i.e. similarity of only 20% - 30%, the *exact* solution was found in about 40% of the cases. Even when the exact permutation was not found, the location of calculated permutation is usually close to the location of the exact permutation. For proteins that differ by 70% - 80%, the average distance between the calculated permutation and the exact one is only about 5 residues.

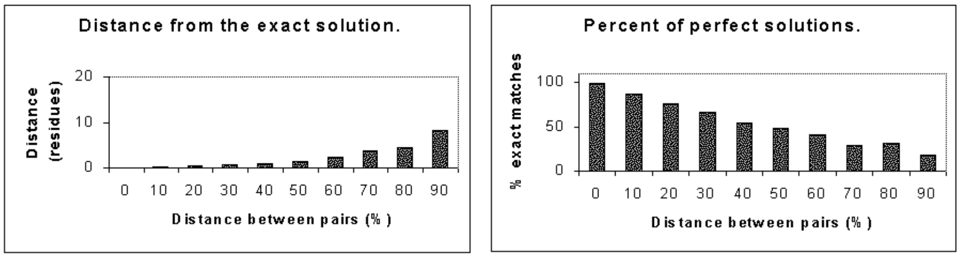

Figure 5: Random cyclic permutations of the protein 411_CHICK, followed by various degrees of random standard mutations were performed. The performance of the algorithm as measured by the percentage of the cases, for each similarity level, for which the correct circular permutation was found (right), and the average distance (in residues) of the found permutation from the correct one in the case of the protein 411_CHICK. In the basic case, were only cyclic permutations were done, the correct results are always retrieved. The performance gradually drops as more mutations are being added. Still, a good level of detection is evident for sequences that have diverged significantly.

4. DISCUSSION

Here we describe how the proposed algorithm can be used as the "main engine" of a large scale survey of circular permutations in the protein database. The algorithm can be used to detect candidate pairs of proteins that underwent circular permutation between all pairs of proteins from the Swissprot database. Since proteins of very different sizes are not probable candidates for a CP event, one does not need to compare proteins whose sizes are significantly different. By restricting the comparison to proteins whose sizes differ by 50%, which is still a wide margin, only about 25need to be scanned. This would require about 2 months of CPU on a single processor machine or a week CPU time on a server like Silicon Graphics Origin 2000 with 8 processors.

As mentioned above (see Figure 4) a significant minimum in the results of the algorithm, is indicative of a circular permutation. Once a reasonable threshold for significant minimum is defined, a list of candidate pairs will be produced. For these pairs, the N^3 exact algorithm will be run to validate that the signal is real. Our preliminary data shows that only a very small fraction (less than 0.1%) of pairs pass the first filtering and need the exact algorithm verification.

A minimum in the edit distance results under circular permutations does not immediately imply that an evolutionary event of circular permutation indeed took place. Thus, for each pair of candidate proteins a few additional tests must be performed. Note that the algorithm produce as a by product the position of the circular permutation (i.e. the location of the minimum). Thus, a comparison can be made between the results of standard sequence alignment procedure performed between the pairs of proteins before and after the permutation. A large change in the Z scores for the significance of the alignments indicates that indeed the proteins are related by circular permutations.

In addition, a visual interpretation of the dotplot matrices, again before and after the circular permutation, can be performed. An indication for circular permutation is when short diagonals which are off the main diagonal before the permutation, will merge together along the main diagonal after the permutation. The dotplot is a good tool to eliminate cases where the original proteins include internal repeats which can give false positive in the analysis. Work in this direction is currently being done by our group.

Note that we are focusing specifically on cyclic permutations. Other types of permutations or domain shuffling (Nalefski and Falke, 1996) which might be common in protein evolution require different tools and are part of a parallel study we are currently conducting.

5. REFERENCES:

Altschul,S.F., Gish,W., Miller,W., Myers,E.W. and Lipman,D.J. (1990) Basic local alignment search tool. J.Mol.Biol., **215**, 403-410.

Ay,J., Hahn,M., Decanniere,K., Piotukh,K., Borriss,R. and Heinemann,U. (1998) Crystal structures and properties of de novo circularly permuted 1,3-1,4-beta-glucanases. Proteins, **30**, 155-167.

Cunningham,B.A, Hemperley,J.J., Hopp,T.P. and Edelman,G.M. (1979) Favin versus concanavalin A:circulary-permuted amino acid sequences. Proc.Natl.Acad.Sci. USA, **76**, 3218-3222.

Goldenberg,D.P. and Creighton,T.E. (1983) Circular and circularly permuted forms of bovine pancreatic trypsin inhibitor. J.Mol.Biol., **165**, 407-413.

Heinemann,U. and Hahn,M. (1995a) Circular permutation of polypeptide chains: implications for protein folding and stability. Prog. Biophys. Mol. Biol., **64**, 121-143.

Heinemann,U. and Hahn,M. (1995b) Circular permutations of protein sequence: not so rare?. TIBS., **20**, 349-350.

Hemperly,J.J. and Cunningham,B.A. (1983) Circular permutation of amino acid sequences among legume lectins. TIBS., **8**, 100-102.

Hennecke,J., Sebbel,P. and Glockshuber,R. (1999) Random circular permutation of DsbA reveals segments that are essential for protein folding and stability. J.Mol.Biol., **286**, 1197-1215.

Landau,G.M., Myers,E.M. and Schmidt J.P. (1998) Incremental string comparison. Siam Journal on Computing, **27**, 557-582.

Lindqvist,Y. and Schneider,G. (1997) Circular permutations of natural protein sequences: structural evidence. Curr.Opin.Struct.Biol., **7**, 422-427.

Luger,K., Hommel,U., Herold,M., Hofsteenge,J. and Kirschner,K. (1989) Correct folding of circularly permuted variants of a beta alpha barrel enzyme in vivo. Science, **243**, 206-210.

Nalefski,E.A. and Falke,J.J. (1996) The C2 domain calcium-binding motif: structural and functional diversity. Protein Sci., **5**, 2375-2390.

Needleman,S.B and Wunsch,C.D. (1970) Ageneral method applicable to search for similarities in amino-acid sequence of two proteins. Journal of Molecular Biology, **48**, 443-453.

Ponting,C.P. and Russell,R.B. (1995) Swaposins: circular permutations within genes encoding saposin homologues. TIBS., **20**, 179-180.

Russell,R.B. and Ponting,C.P. (1998) Protein fold irregularities that hinder sequence analysis. Curr.Opin.Struct.Biol., **8**, 364-371.

Smith,T.F. and Waterman,M.S. (1981) Identification of common molecular subsequences. Journal of Molecular Biology, **147**, 195-197.

Unger,R., Harel,D. and Sussman,J.L. (1986) DNAMAT:an efficient graphic matrix sequence homology algorithm and its application to structural analysis. CABIOS., **2**, 283-289.