International Journal of Computer Applications (0975 – 8887)
Volume 50 – No.12, July 2012

A Multi-core Tool for Searching Protein Structural Similarities

Ahmed Salah
College of Information Science and Engineering,
Hunan University
Faculty of Computers and Informatics,
ZagazigUniversity

KenliLi
College of Information Science and Engineering,
Hunan University

Tarek F. Gharib
Faculty of Computing and Information Technology,
King Abdulaziz University,Saudi Arabia
Faculty of Computer and Information Sciences, Ain Shams University

AbdulFattahMashat
Faculty of Computing and Information Technology,
King Abdulaziz University,Saudi Arabia

ABSTRACT

The analysis of protein structural similarities plays an important role in different biological fields. These fields vary from the process of developing new drugs to detecting the evolutionrelationships. As the number of protein structures grows rapidly there is an increasing demand for improving the speed of the computational tools that handles proteome. The wide prevalence of multi cores computers and its low price can be employed to speed up the existing tools used for searching protein structural similarities. In this report, we present a modified version of a PSISA tool, which efficiently used to find the structural similarities between different proteins and maintains the load balance between cores. Using an Intel 8 cores computer and the structural classification of proteins (SCOP) dataset, the experiments show an average speed up 1.8 using 8 cores without affecting the memory usage or the accuracy of the tool.

General Terms

Algorithms, Bioinformatics, Computational Biology.

Keywords

Protein Structural Similarities, Multi-core, Structure Comparison, Indexed protein structure, suffix array.

1. INTRODUCTION

The function of proteins depends on both its sequence and structure [1]. Protein structural similarity comparison is a key player in the process of drug design and other several vital biological fields. Each protein has a file that contains the coordinate of each amino acid in the space. So we have a set of proteins with known functions and this set can be used as the database to predict and understand the function of the new discovered proteins. For any query protein the input should be its amino acids 3D coordinates, and the result of any method, intended to find structure alignment, should be a list of the known protein with the highest structural similarities.

Abundant of methods have been established to perform the protein structure searching. That method can be classified into three classes. The first class performs one to one comparison by comparing each entry of the database with the query protein as suggested in [2], CE [3], and DALI [4]. Recently, an iterative method is proposed to handle specifically the structure similarity in distantly related proteins with low sequence identity [5]. The second class performs a structural alignment which produces a set of superposed three-dimensional coordinates for every query proteins at the residue level [6]. A mathematical framework for protein structure comparison based on elastic shape analysis is proposed. Under this framework, protein structures are compared as three dimensional elastic curves and can be treated as random variables for statistical analysis; this framework is not good for detecting related proteins with differences are caused by changes such as domain insertion/deletion or domain swapping [7]. Recently, a method proposed a graph alignment as a mean for comparing proteins on a structural level. This method is a semi-global strategy because it shares properties with both local and global graph matching, similar to semi-global sequence alignments [8].These two classes produce very highly accurate results but in terms of processing time they require long running time, the searching process can cost hours to days. The third class is indexing protein structure based on protein backbone three-dimensional coordinate values [9, 10] or environment information [11]. The third class is focusing on local similarity rather than global similarity that affect the overall accuracy but results are still comparable, meanwhile thisclassis significantly faster than the former two classes. To conclude, it is efficient, third class, against accuracy, first and second classes.
Due to the rapidincreasing of discovering new proteins and the wide prevalence of multi core computers, we focus in this report to design a tool which is a time and memory efficient for protein structure comparison. We provide the efficiency by selecting the one approach belongs to the third class mentioned above and then to implement the multi-core version. PSISA is a time and memory effective algorithm presented to approach the protein structure comparison problem [12, 9, 13]. The reason for selecting PSISA algorithm to be in a multi core version, that it is based on suffix arrays as the indexed structure [14]. This data structure is suitable for parallelizing and it is famous ofits memory efficiency. Meanwhile other tools are difficult to be parallelized like PSIST [9] since it is based on trees as the indexing structure. Also the presence of parallel algorithms for building suffix arrays is another point for selecting the PSISA.

In this research, we present the first multi-core version of algorithms belongs to the third class. We have selected an efficient algorithm in terms of memory and speed. Testing this tool, we gain a speedup of 1.8 using 8 cores without affecting the memory usage or the accuracy of the tool.

2. SOFTWARE INPUT/OUTPUT

Multi-core PSISA (MPSISA) is a tool developed in Java in order to be a cross platform tool. There are no advanced requirements to run the tool just the javarun time environment (JVM) which is freely available. The basic idea of the tool to have a set of proteins with known function this set considered as the database set, and another set of proteins with unknown functions which is the query set. The tool starts searching the database set in order to find proteins similar to the proteinsof the query set.

The input for MPSISA is a path that contains all database proteins to be indexed and the path for the query proteins. The list of complete MPSISA input parameters is listed in Table 1. The *w* parameter is used to determine the sliding window size, as *the w parameter* increases the required memory for the program increases. The *b* parameter is used to normalize the feature vector values extracted as it increases the accuracy and running time increase. The *l* parameter determines the threshold to consider the match, any match with length less than *l*value is not counted as it *l*they accuracy and running time increase. The *dbdir*and *querydir*parameters determine the directory for database set and query set respectively.

The output for each query protein is a list of proteins, from the database set, that match the query protein. The list is sorted by the matching scores.

Table 1. MPSISA Parameters

Parameter	Valid values	Meaning	Default
W	Integer > 1	Window size	3
B	Integer > 1	Bin value	2
L	Integer > 1	Matched length threshold	15
Dbdir	Directory Path	Pathto the database set directory	n/a
Querydir	Directory Path	Pathto the query set directory	n/a

3. EXPERIMENTAL RESULTS

We designed an experiment to test the speedup of the query time, which is the main target of the improvement in this research. To test our tool we selected the SCOP dataset. In SCOP, 621 is the number of super families which contain more than 4 proteins. We have built the known protein database by using at most 5 proteins from each family thatproduced 3105 proteins which presents the database. In order to build the query database we have selected one protein from each super family that selected proteins has the longest data.The resulted 621 proteins present the query database. We then run the program once for each query protein and calculated the running time. The results of the speed uppresented in the following are the average running time of these 621 runs.

Figure1 shows an increment in the speedup as the number of cores increases. The usage of 8 cores achieves an average speedup equals to 1.5 for the 621 queries selected from different protein super families.

Figure 2 shows the relation between the speedup of the query time and the query file size, as the protein size increases the query file increases. We can conclude that there is no relation between the size of the query and the speed up, the speedup of query time depends mainly on parallelizing the elements of the query suffix array and a database suffix array, so the speed up increases as the number of query suffix array increases which has no connection with the size of the query protein.

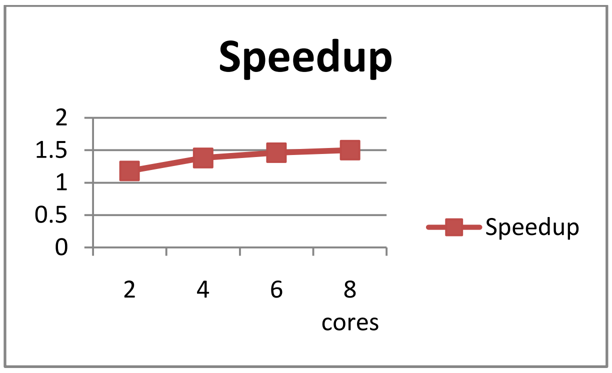


Fig 1: The speedup of PSISA algorithm using up to 8 cores

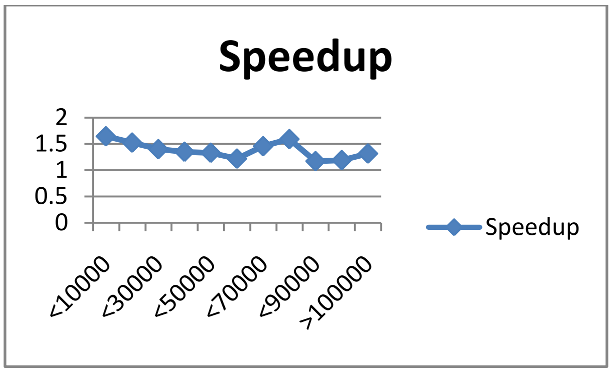


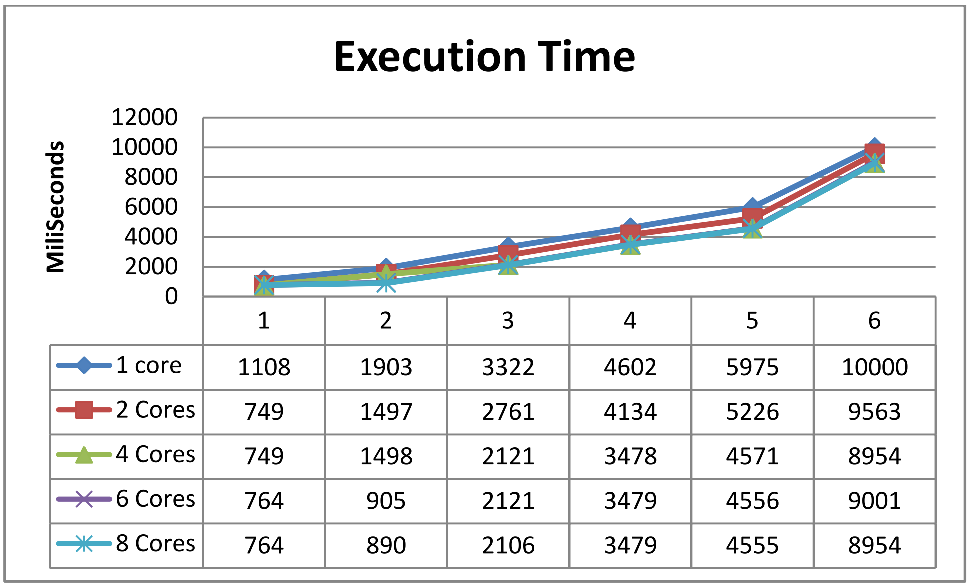
Fig 2: The relationship between speed up and file size

Figure 3 shows the performance of using different number of cores to response to the query protein. Forrandomly selected 6 proteins of ascending execution time for the single core, the execution time is shown for 1, 2, 4, 6, and 8 cores. The figure shows that execution time of 4, 6, and 8 cores is very close,meanwhile the values of execution times for 8 cores outperform the2,4, and 6 cores' execution times slightly.

4. FUTURE WORK AND CONCLUSION

The rapidly increasing in the number of discovered proteins should be considered as a challenge for a single CPU computer. To overcome this challenge a cluster-based version of MPSISA is highly required.Achieving this task increases MPSISA scalability so that it can handle huge amount of protein structure comparison.

In this paper, we propose a parallel toolfor protein structure comparison, which is based on existing sequential tool. This tool is developed to overcome the slow performance due to the increasing in protein structures number. This proposed methods speed up by 1.8 times on average using 8 cores. Thespeed up rate depends on the datasetsize, as the datasetsize increases the speed up rate increases.



	1	2	3	4	5	6
◆ 1 core	1108	1903	3322	4602	5975	10000
■ 2 Cores	749	1497	2761	4134	5226	9563
▲ 4 Cores	749	1498	2121	3478	4571	8954
× 6 Cores	764	905	2121	3479	4556	9001
✱ 8 Cores	764	890	2106	3479	4555	8954

Fig 3: The performance comparison between the different cores

5. REFERENCES

[1] Lee, D.,Redfern, O., and Orengo, C.2007. Predicting protein function from sequence and structure. Nat Rev Mol Cell Biol.

[2] Eidhammer,I., Jonassen, I., and Taylor, W. R.2000. Structure Comparison and Structure Patterns. Journal of Computational Biology.

[3] Shindyalov, I. N., and Bourne, P. E. 1998. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. Protein Engineering.

[4] Holm, L.,Kääriäinen, Rosenström, S.P., andSchenkel, A. 2008. Searching protein structure databases with DaliLite v.3. Bioinformatics.

[5] Venkateswaran,J. G., Song, B.,Kahveci, T.andJermaine, C. 2011. TRIAL: A Tool for Finding Distant Structural Similarities. IEEE/ACM Transactions on Computational Biology and Bioinformatics.

[6] Konagurthu,A. S.,Whisstock, J. C., Stuckey, P. J., and Lesk, A. M. 2006. MUSTANG: a multiple structural alignment algorithm. Proteins.

[7] Liu, W.,Srivastava, A., and Zhang, J. 2011. A Mathematical Framework for Protein Structure Comparison. PLoS Computational Biology.

[8] Mernberger, M., Klebe, G., and Hullermeier, E. 2011. SEGA: Semiglobal Graph Alignment for Structure-Based Protein Comparison. IEEE/ACM Transactions on Computational Biology and Bioinformatics.

[9] Gao,F., and Zaki,M. J. 2008.PSIST: A scalable approach to indexing protein structures using suffix trees. Journal of Parallel and Distributed Computing.

[10] Shibuya,T. 2004.Generalization of a Suffix Tree for RNA Structural Pattern Matching.Algorithmica.

[11] Carpentier,M.,Brouillet,S., and Pothier,J.2005. YAKUSA: a fast structural database scanning method. Proteins.

[12] Gharib, T. F.,2009. A hybrid approach for indexing and searching protein structures. W. Trans.onComp.

[13] Gharib, T. F., Salah, A., and Abdel-Badeeh, M. S. 2008. PSISA: an Algorithm for Indexing and Searching Protein Structure using Suffix Arrays. WSEAS International Conference on COMPUTERS.

[14] Manber,U.andMyers, G. 1990. Suffix arrays: a new method for on-line string searches. InProceedings of the first annual ACM-SIAM symposium on Discrete algorithms.