American Journal of Applied Sciences 6 (7): 1368-1372, 2009
ISSN 1546-9239

The Parallel Maximal Cliques Algorithm for Protein Sequence Clustering

Khalid Jaber, Nur'Aini Abdul Rashid and Rosni Abdullah
School of Computer Sciences, University Sains Malaysia, 11800 Penang, Malaysia

Abstract: Problem statement: Protein sequence clustering is a method used to discover relations between proteins. This method groups the proteins based on their common features. It is a core process in protein sequence classification. Graph theory has been used in protein sequence clustering as a means of partitioning the data into groups, where each group constitutes a cluster. Mohseni-Zadeh introduced a maximal cliques algorithm for protein clustering. **Approach:** In this study we adapted the maximal cliques algorithm of Mohseni-Zadeh to find cliques in protein sequences and we then parallelized the algorithm to improve computation times and allowed large protein databases to be processed. We used the N-Gram Hirschberg approach proposed by Abdul Rashid to calculate the distance between protein sequences. The task farming parallel program model was used to parallelize the enhanced cliques algorithm. **Results:** Our parallel maximal cliques algorithm was implemented on the stealth cluster using the C programming language and a hybrid approach that includes both the Message Passing Interface (MPI) library and POSIX threads (PThread) to accelerate protein sequence clustering. **Conclusion:** Our results showed a good speedup over sequential algorithms for cliques in protein sequences.

Key words: Maximal cliques algorithm, parallel processing, protein sequence clustering, MPI, pthread

INTRODUCTION

One of the basic applications of protein sequence comparison is in protein sequence clustering. Protein sequence clustering is an element of protein sequence analysis. The results of protein sequence clustering can be used as a basis for the prediction of new protein sequence structure and function, or as a basis for protein sequence classification. The two basic steps to protein sequence clustering include calculating distances among the protein sequences and grouping the sequences into groups of similar sequences based on these distances. The N-Gram-Hirschberg technique[2] is used to calculate the pairwise distance between a pair of sequences. The resulting distance values are stored in a distance matrix. We used a clustering algorithm based on a maximal clique proposed by Mohseni-Zadeh *et al*.[1]. Maximal cliques are used to find a cluster in a set of protein sequence graphs. However, we adapted the algorithm to find cliques of different sizes using the graphs. Relationships between protein sequences can readily be shown on a graph. Nodes or vertices in the graph represent protein sequences while each edge represents a relation between two vertices. A clique is a subset of vertices, such that all vertices in the subset are directly connected to each other. The out-degree of each vertex is (n-1), where n is the number of vertices in the subset.

The cliques algorithm is an extension of a large scale clustering algorithm that is based on extracting maximal cliques[1]. There are three steps involved in finding cliques:

- Search for a Maximal Clique (MC) that is a core cluster. Other nodes that do not meet the clique criteria are placed in a set of Non-Cliques (NC)
- Extend the maximal cliques by finding all the nodes related to any of the core clusters
- Find more sequences that are linked to the new nodes just added to the core clusters in step 2
- Repeat steps 1-3 to find additional cliques

In this study, we extend this study to find multiple maximal cliques and we apply the Parallel Maximal Cliques Algorithm (PMCA) on the protein sequences taken from various protein databases. Our algorithm has been implemented by[2] in a single-processor computer system. The framework for clustering protein sequences is shown in Fig. 1 and the sequential algorithm for finding cliques is shown in Fig. 2.

Corresponding Author: Khalid Jaber, School of Computer Sciences, University Sains Malaysia, 11800, Penang, Malaysia
Tel: +604-6533888 Fax: +604-6573335

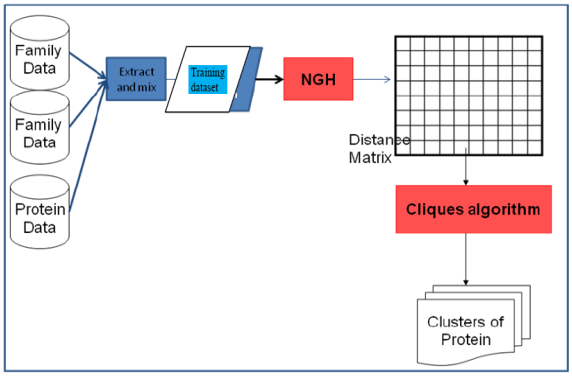

Fig. 1: Steps in clustering protein sequences[2]

Finding Cliques in a graph
Input: Directed Graph, t: clique threshold value

1. Sort the graph head nodes list by descending degree of node.
2. For all H head nodes in graph head nodes list
   a. For all n nodes in H
      i. Find n in the Head nodes list
      ii. If all nodes connected to n exist in H then delete n from the head graph nodes list.
         1. If the number of nodes connected to n > cliques threshold value
            a. Insert the nodes connected to n into H.
            b. Delete n from H.
      iii. Else delete n from H.

Fig. 2: Finding cliques in a graph[2]

The sequential implementation took a long time to process because of the huge data volume[2]. This will continue to pose a major challenge in the future. Our research focuses on parallelization instead of implementing the algorithm on a single machine. One of the main advantages of the maximal cliques technique is that it is inherently parallelizable. The choice to implement the Parallel Maximal Cliques Algorithm on a cluster was based on the fact that workstation clusters are cheap and readily available. Clusters can also be easily expanded and have low maintenance costs. Moreover, development tools on workstations are mature.

This study offers three principal contributions. First, we present a parallel processing design for the Maximal Cliques Algorithm. Second, we apply this approach to protein sequencing; and finally, we parallelize the maximal cliques algorithm.

This study is organized as follows. Describes the proposed design then addresses the implementation results and discussion. Finally we offer conclusions.

Proposed design: Our purpose is to design a parallel version of the maximal cliques algorithm for protein sequence clustering. We will discuss the design from many perspectives such as the parallel maximal cliques algorithm model, hardware and software, number of processors and lastly input and output.

PMCA models: the Parallel Maximal Cliques Algorithm (PMCA) is implemented using a Task Farming (or Master/Slave) model.

The master supervises the pre-processing phase in which the number of files (FASTA protein files) to be placed in the structure is read. This is our project database. Then the system reads threshold values and clique thresholds. It extracts information from the FASTA file and saves it as a number of structures. Subsequently, it computes the relation between two protein sequences (using N-Gram Hirschberg Algorithms). Relations can be similarity values or distance values. The distance values are stored in a similarity matrix (Fig. 3). The relation between two protein sequences is implemented using POSIX threads (PThread). The similarity matrix is partitioned and distributed into different threads. The threads are forced to run on different processors. Threaded programs are significantly easier to write because threaded applications that run on a single machine can subsequently run on multiple machines without changes. This ability to migrate programs between different platforms is a great advantage for threaded APIs.

The master node's primary responsibility is to distribute the protein sequences (jobs) among the slaves. The program can run on two, three and five processors. Protein sequences are assigned to slave nodes based on the total number of sequences (i.e., if we have 20 sequences and 3 nodes (1 master, 2 slaves), the master sends 10 sequences to each slave node).

When the master node receives the results from the slaves, it aggregates and processes the final results. In summary, the master's responsibilities are:

- Carry out the pre-processing phase (read FASTA protein files, threshold value and clique threshold)
- Compute relations between two protein sequences (using N-Gram Hirschberg Algorithms)
- Decompose the problem into smaller tasks
- Distribute the protein sequences among the slaves
- Gather results from the slaves and process final conclusions

The slave node's primary responsibility is to build a directed threshold Graph Gt that includes values greater than a given threshold t. Subsequently the slave sorts the graph head nodes in descending order.

It then finds all the cliques in the graph and transmits the result to the master node.

In summary, the slave's responsibilities are:

- Accept task from master
- Process the task (build graph and sort it)
- Find all cliques in the graph
- Transmit results to master

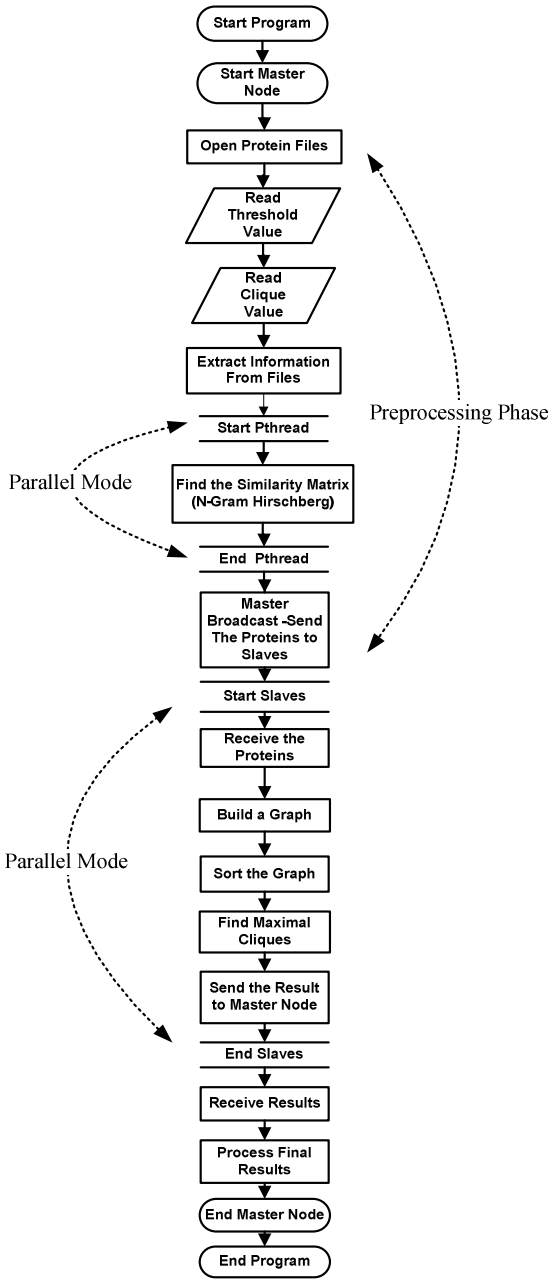

Fig. 3: PMCA flow chart

MATERIALS AND METHODS

Experimental environment: The Parallel Maximal Cliques Algorithm is implemented on the Stealth cluster using the C programming language and a hybrid of the Message Passing Interface (MPI) library and POSIX threads (PThread), as mentioned previously. The Stealth cluster consists of 5 machines where one machine is the server node and the other four are child node. The Stealth cluster is located at the Parallel and Distributed Computing Lab at the School of Computer Sciences, USM. The configuration of the Stealth cluster is shown in Table 1.

The cluster is interconnected by fast ethernet. One limitation of the cluster is that it is shared by many users around campus, so result can be unreliable when the load of the machine is high. To get appropriate usable result, we thus tested the program during the middle of the night and ran our test protocol several times before averaging the results.

RESULTS AND DISCUSSION

The protein sequence data was taken from experiments that examined protein sequences used by[2]. We tested the program on four data sets extracted from public domain databases. The first dataset (dataset1) was used by[3] and is denoted COG 001. 0160 was taken from[4]. The first dataset consists of 114 protein sequences.

The second dataset (dataset2) is derived from four PFAM families. It consists of 212 protein sequences. The third Dataset (dataset 3) consists of 319 protein sequences from NCBI and the fourth Dataset (dataset 4) consists of 295 families from Swiss-Prot.

The second, third and fourth datasets were downloaded from Protein Information Resources (PIR) at[5]. All the data used were in FASTA format. Table 2 shows details of each dataset.

Table 1: Stealth cluster configuration

Compute nodes	1 Master node
	4 Slaves node
Hardware	Master Node:
configuration	Sun Fire 280R:
	2 × Sun Sparc III 900 MHc processor
	2 GB RAM
	4 network interface cards
	Other Nodes (slaves)
	Sun Fire v10
	2 × Sun Sparc III 900 MHz processor
	2 GB of RAM
	1 network interface cards connected using fast ethernet
Operating system	Sun solaris 9
and software	Sun biobox
	Sun HPS cluster tools 5.0
	MPI, Gnu C

Table 2: Datasets

Data set	Source	No. of protein sequence
1	COG0001_00160	114
2	PFAM1	200
3	NCBI	319
4	Swiss-prot	295

Table 3: Performance measures for dataset 1

No. of processor	Time (sec)	Speedup	Efficiency
One	297.71		
Two	169.26	1.75	0.87
Three	149.37	1.99	0.66
Five	146.99	2.02	0.40

Table 4: Performance measures for dataset 2

No. of processor	Time (sec)	Speedup	Efficiency
One	355.43		
Two	205.94	1.72	0.86
Three	185.48	1.91	0.63
Five	180.32	1.97	0.39

Table 5: Performance measures for dataset 3

No. of processor	Time (sec)	Speedup	Efficiency
One	4066.01		
Two	2177.31	1.86	0.93
Three	2169.12	1.87	0.62
Five	1868.74	2.17	0.43

```
>PF00181|NF01243182 50S ribosomal protein L2 [Coxiella
burnetii]
MALVKTKPTSPGRRFVVKVVHPELHKGDPYAPLVESKNR
INSRNNQGRITVRRRGGGHKRNYRIIDFKRDKEGIEGKVE
RLEYDPNRSAHIALVLYPDGERRYIIAPKGVHKGSKVVSG
REAPIRPGNCLPLQNIPLGATIHNIELKPGKGAQLVRSAGA
SAQLAAKEGIYAIIRMRSGETRKILAVCRACIGEVSNSEHN
LRSLGKAGAKRWRGRRPTVRGVAMNPVDHPHGGGEGK
TSGGRHPVSPTGKPTKGYKTRRNKRTSNMIIRDRRKK
```

Fig. 4: Example of input data in FASTA format from PIR

Each data sequence was labelled according to the PFAM number, which denotes the family in which the protein sequence belongs. This label assumes the existence of a "true" cluster. For example, if the protein sequence comes from PIR and features a PFAM number of PF00181, then the tag of each protein sequence in the FASTA format starts with the family number. Figure 4 shows one such entry.

The sequential Maximal Cliques Algorithm took about 297.71 sec to process dataset1, 355.43 sec for dataset 2, 4066.01 sec for dataset 3 and 1392.94 sec for dataset 4 (Table 3-6). We implemented the parallel version on two, three and five processors. Detailed results are shown in Table 3-6.

Table 6: Performance measures for dataset 4

No. of processor	Time (sec)	Speedup	Efficiency
One	1392.94		
Two	840.02	1.65	0.82
Three	782.13	1.78	0.59
Five	673.03	2.06	0.41

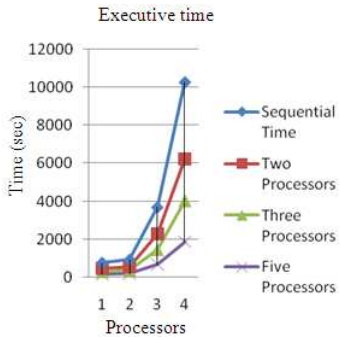

Fig. 5: Performance results

The purpose of executing our program on a single processor was to assess sequential running time and to subsequently use it as a benchmark against which to compare parallel running times. Our subsequent results allowed us to calculate both performance gain and efficiency of our parallel program.

The execution time required to process datasets 1-4 using PMCA was less than that required when using MCA, as shown in Table 3-6. However the accelerated PMCA operated faster when implemented on two processor nodes. Based on the experiments and results shown in Table 3-6 and Fig. 5. But we noted no significant difference between the execution time on three and five processors because inter-processor communication time increased and this negatively impacted MPI, which is highly communication-dependent.

CONCLUSION

As the volume of biological data continues to increase exponentially, parallel systems are urgently needed to help scientists accelerate their analyses. This can be achieved by enhancing the protein sequence clustering algorithm. The parallel maximal cliques algorithm is the most widely-used method for protein sequence clustering, which constitutes the basic method for discovering relations between proteins.

In this research, we have applied parallel methods to improve the computational time required to identify clusters in large biological protein data sets. The parallel maximal cliques algorithm has been parallelized on two levels. The first level used POSIX

threads (PThread). At this level, the similarity matrix was partitioned and distributed to different threads. The threads were force to run on different processors. The Message Passing Interface (MPI) is the second level of parallelism. MPI is used as the message passing library to allow inter-processor communication. We conclude by hybridizing the two levels of parallelism.

Our results from running the parallel algorithm have shown good acceleration and efficiency in two-processor tests. However, we observed no significant difference between the execution time on three vs. five processors, because the time devoted to communication between processors increased and this exposed the main weakness of MPI, which is highly dependent on message parsing. In the future, we hope to run the experiments on GPGPU (general-purpose computing on graphics processing units) to obtain more conclusive results with large data sets.

ACKNOWLEDGMENT

We would like to thank Najwa Abu Bakar at Universiti Sains Malaysia for fruitful collaboration opportunities. This research was funded by grant titled "Parallel Graph-based Algorithm for Protein Tertiary Structure Matching" [01-01-05-SF0431] .

REFERENCES

1. Mohseni-Zadeh, S., P. Brezelec and J.L. Risler, 2004. Cluster-C, an algorithm for the large-scale clustering of protein sequences based on the extraction of maximal cliques. Comput. Biol. Chem., 28: 211-218. http://www.ncbi.nlm.nih.gov/pubmed/15261151
2. Abdulrashid, N.A., 2008. Enhancement of hirschberg algorithm using N-gram and parallel methods for fast protein homologous search. PHD Thesis, School of Computer Sciences, University Sains Malaysia.
3. Kim, S. and J. Lee, 2006. BAG: A graph theoritic sequence clustering algorithm. Int. J. Data Min. Bioinform., 1: 178-200. http://www.ncbi.nlm.nih.gov/pubmed/18399070
4. Bioinformatics, Aprial 2008. http://bio.bioinformatics.indiana.edu/sunkim/BAG/
5. Protein Information resource, 2008. http://pir.georgetown.edu/