Alternative Parallelization Strategies in EST Clustering

Nishank Trivedi[1], Kevin T. Pedretti[2], Terry A. Braun[1], Todd E. Scheetz[1], and Thomas L. Casavant[1]

[1] The University of Iowa, Iowa City,
Iowa 52242, USA,
tomc@uiowa.edu
[2] Sandia National Labs, Albuquerque, New Mexico,
87123, USA

Abstract. One of the fundamental components of large-scale gene discovery projects is that of clustering of Expressed Sequence Tags (ESTs) from complementary DNA (cDNA) clone libraries. Clustering is used to create non-redundant catalogs and indices of these sequences. In particular, clustering of ESTs is frequently used to estimate the number of genes derived from cDNA-based gene discovery efforts. This paper presents a novel parallel extension to an EST clustering program, `UIcluster4`, that incorporates alternative splicing information and a new parallelization strategy. The results are compared to other parallelized EST clustering systems in terms of overall processing time and in accuracy of the resulting clustering.

1 Introduction

The sequencing of cDNA libraries is the most common format for gene discovery in higher eukaryotes. The goal of such a project is to utilize the sequences derived from the cDNAs (ESTs; expressed sequence tags) to derive a non-redundant set. This set ideally represents an organisms entire complement of genes. EST-based gene-discovery projects are in progress for numerous species of medical, scientific, and industrial interest. The benefits of EST-based gene discovery include the ability to rapidly identify transcribed genes, the ability to identify exon-intron structure (when coupled with genomic sequence), and information on gene expression. EST data is so useful that the National Center for Biotechnology Information (NCBI) provides a separate division specifically for EST sequences (dbEST) [1].

However, different genes are expressed at different levels. Thus, a given gene's transcript may be present in 0, 1 or many copies within a cell. Because these transcripts are used to generate the cDNAs, both the cDNAs and the ESTs derived from them will also be present in similarly variable levels.

The presence of this redundancy within the EST databases, requires a programmatic method to calculate the complement of genes they represent. These methods (termed clustering) utilize sequence-based comparisons to determine

V. Malyshkin (Ed.): PaCT 2003, LNCS 2763, pp. 384–393, 2003.

sets of strongly similar sequences (clusters). The primary difficulty associated with EST-based gene-discovery projects is that ESTs are single-pass sequences, and as such they are relatively error prone (approximately 3% on average [2]).

NCBI also provides a curated and annotated gene index (UniGene) [3] for several species, utilizing the available mRNA and EST sequences to estimate the gene complement. This paper describes and compares several programs that may be used to create non-redundant "UniGene" sets from EST data, and analyzes three different approaches for parallelization of this task.

2 Background

Clustering plays an important role in large scale gene-discovery projects. It not only saves time by identifying redundant (EST) sequences but also provides useful information regarding gene-discovery rate [4]. Another significant use of clustering is to create non-redundant gene indices. As suggested in this paper, clustering can be further used to identify possible alternatively spliced sites in mRNA gene transcripts

There are several varying clustering methods and tools in use today. However, the objective of all such methods is to effectively assess the similarity between all pairs of sequences and place them into equivalence classes. Ideally, these classes correspond, one to one, onto distinct genes. It should be noted that such a procedure based entirely on subsequence similarity cannot achieve perfect fidelity with respect to gene classes. A number of other criteria, not apparent in the primary RNA sequence data, are necessary for such a classification. However, a sequence-based classification is of extremely high usefulness. One of the most widely used clustering tools is NCBI's Unigene clustering [5]. It uses global pairwise sequence comparison, and a stringent protocol for assigning closely related sequences to a common cluster. However, it does not support incremental clustering. Hence, each clustering "build" must begin from the same initial starting point. As the number of known ESTs in homo sapiens currently stands at approximately 5 million, and requires more than one month of computation time, the ability to perform incremental clustering becomes obvious. Also, an EST relating to two different clusters is discarded, overlooking any possible alternative splice sites. The Institute for Genome Research (TIGR) [6], produces gene indices for many organisms. It performs a pairwise alignment of incoming sequences with a template obtained from a database consisting of expressed mRNA transcripts, as well as tentative consensus assemblies of other ESTs, mRNAs and cDNAs. Sequences must qualify through strict identity criteria. Each cluster is finally assembled to produce a consensus sequence. Due to very strict clustering rules, TIGR gene indices discard many under-represented, divergent or low-quality sequences, leading to under-clustering of sequences. The SANBI STACK clustering approach [7] was developed primarily for human databases, but is general purpose. It performs a looser clustering of sequences, but has a strict assembly phase. The clustering is conducted using non-contextual assessment of composition, and a multiplicity of words within each sequence. Typically, the STACK

approach produces larger clusters than Unigene, and has longer consensus sequences for each cluster than TIGR.

All of the above programs are essentially sequential. A parallel clustering method developed at Iowa State University [8], `PaCE`, uses an implementation of suffix trees for sequence comparison. The method is a strict sequence identity match clustering method and performs an NxN alignment, although in parallel, hence offsetting the high costs associated. The overall method is to construct suffix trees in parallel, perform pairwise alignment for selective sequences, and finally to group them together based on a similarity score. However, the method demands a specific hardware requirement, and overlooks divergent cases within EST sequences. The `UIcluster` family of solutions (both serial and parallel) has been evolving in a production environment at the University of Iowa since 1997. The key characteristics of `UIcluster` are incremental clustering, the maintenance of a "primary" representative element for each cluster, and a hashing scheme to quickly identify potentially meaningful cluster matches for each newly considered input sequence. The stringency of the clustering is a user-definable parameter, although performance is also a sensitive function of this aspect. Parallelization of `UIcluster` has now been performed relative to both the cluster space [9], and input space, which is the main focus of this paper.

The following is a brief description of the underlying approach of `UIcluster`. After an incoming sequence has been read from an input file, it is compared against all existing clusters. The comparison is performed only with the primary element of each cluster, where the primary is a single representative sequence of the entire cluster (usually the longest, and therefore most informative member). If the incoming sequence matches, or "hits" any cluster primary, and further satisfies specified similarity criteria, it is added to that cluster. Otherwise, the incoming sequence itself becomes the primary element of a new cluster. The basic search screening is based on a criteria of n matching positions in a window of length m. Hashing is usually performed on a short motif roughly one quarter of the size of m. The $m - n$ positions in discordance may be either substitution or gap errors. As a practical necessity, a global table of hash values and a map to each cluster containing those values, is used to count hits to the primaries of any given cluster. In many cases, it is possible to avoid actual alignment of a new sequence to the primary of the best cluster hit by employing deduction based on the number of hashes which correspond between the two sequences. The efficiency gained by using hashes to a primary, and thus avoiding alignment is dramatic. Basing our parallel versions of `UIcluster` on this already optimized serial method provides a number of advantages to be described in the next section.

3 Approach and Implementation

The performance of EST clustering is measured both by time as well as by memory resource utilization characteristics. Although the use of hashing, and of a primary sequence for each cluster significantly reduces both these requirements,

space remains a limiting factor for even more efficient and heuristic approaches. Considering the nature of the problem and the size of the data set, parallelization is an obvious choice for the implementation of this process. Computational and memory requirements can be distributed across several computers. This adds to the performance of software so that the program can scale to larger problem sizes.

`UIcluster` currently implements two different approaches of parallelization, distributing across the cluster space and the input space. The MPI (message passing interface) [10] standard is used for inter-process communications, and distribution is done among multiple UNIX processes.

3.1 Parallelization on Cluster Space

In this scheme of parallelization, implemented in `UIcluster3`, each cluster is stored on exactly one compute node. The clusters are evenly distributed among all nodes. When a sequence is brought in, it is copied to all available nodes and is processed in parallel. Since each node has a different set of clusters, the incoming sequence is compared with the divided cluster space in parallel. For every node, once the local search has been performed, the information about the best matching primary is communicated to all other nodes. Further, the node with the best match adds the sequence to its cluster space. In the case of a non-match, the sequence itself becomes a cluster and is designated to one compute node.

3.2 Parallelization on Input Space

In the cluster space parallelization method, incorporated in `UIcluster4`, the input is the same for all the compute nodes but the clusters being created are distributed over various processors. A variation on this scheme can be implemented by dividing the sequences also in N non-overlapping groups, where N is the number of compute nodes available. Instead of distributing clusters over different compute nodes and processing each sequence at all nodes, in this scheme each node gets an individual sequence. The pool of input sequences is evenly distributed among all compute nodes. This is similar to running sequential version of `UIcluster` in parallel on all nodes, however, with an abridged dataset. Each node computes its own set of primaries. In the second stage, these primaries are compared among themselves and related clusters are merged.

The efficiency of this scheme is heavily dependent on the redundancy within the dataset. If the data has a high rate of redundancy, the clusters being created on each node are more likely to be merged, involving more communication and added processing. On a single node or cluster space parallelization, the redundant data would have converged into a smaller number of clusters. On the contrary, less redundancy amounts to more clusters hence, input space parallelization reduces space and time requirements.

4 Related Issues

4.1 Virtual Primaries

One limitation in early versions of `UIcluster` was the requirement that a single representative sequence (primary sequence) be selected. Even when mRNA transcript sequences are available, they often lack comprehensive coverage of the original transcript (especially the untranslated regions). Therefore, EST sequences generated from the 3' end may contain significant amounts of novel sequence not represented in the mRNA sequences. Other, more complex, processes such as alternative splicing and alternative polyadenylation are also sources of additional novel sequence. To address this issue, we have developed the concept of a virtual representative sequence (virtual primary). A virtual primary is a non-redundant representation of the constituent sequences within a cluster. Utilizing virtual primaries enables sequence comparisons to be performed against only one sequence per cluster, while still searching the entire composite sequence available for each cluster.

Figure 1 illustrates how a set of partial sequences may be combined to construct a virtual primary. Here, alternate shading is used to denote blocks of homologous sequence. On the left is a set of ESTs (A,B,C,D,E) derived from the same gene. The right half shows the effect of adding the sequences into a growing virtual primary. With a single sequence (A) the virtual primary is identical to the EST. As sequences containing novel subsequences (B,C) are added into the cluster, the novel portions are integrated into the virtual primary at the appropriate position (A+B+C). If a sequence contains no novel subsequences (D), the virtual primary is not changed. In the event of a sequence with a novel insertion (with respect to the virtual primary) (E), the novel portion is incorporated into the virtual primary at the congruent position (A+B+C+D+E).

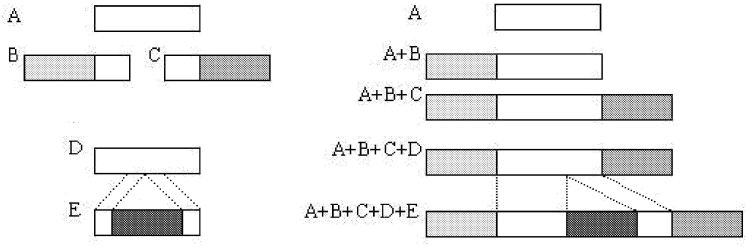

Fig. 1. Contruction of a virtual primary. ESTs derived from transcripts of the same gene are shown at left (A, B, C, D, and E). At right the growing virtual primaries are shown as each EST is included. The dashed-lines represent regions of sequence homology.

Incorporating the construction of virtual primaries into the clustering procedure does not affect the strategy used to identify which cluster a sequence belongs

to. The only impact is to alter the method of deriving the primary sequence. This process does add a small overhead into the computation cost clustering, but does not alter the computational complexity of the algorithm.

4.2 Cluster Viewing and Editing

An ancillary program has been developed to aid in the visualization and editing of the resulting clusters. This cluster editor was implemented in Java as both an application and an applet. The applet-based solution makes our clustering method available over the internet to interested users. Search features were integrated into the editor so that clusters with specific features can quickly be identified. Currently supported features include clusters with apparent alternative splicing, and those with weak sequence hits - potentially from gene families. This program facilitated the process of debugging the clustering program, enabling erroneous cases to be visualized. Two issues in the construction of virtual primaries particularly benefited from the use of the cluster editor: the order in which non-redundant sub-sequences were incorporated, and that all non-redundant sub-sequences are included in the virtual primary.

4.3 Order of Inclusion

One factor that can significantly affect the clustering is the order in which sequences are included. `UIcluster's` approach performs smoothly for short EST sequences (400-1,000bp). However, full length mRNA sequences (1000's of bp) may rapidly degrade the performance. As the sequence length increases, so does the probability of finding additional minimally-matching regions, which results in an increase in the number of detailed sequence comparisons. For `UIcluster3`, splitting the longer sequence into smaller overlapping sequences was a potential solution. However, when `UIcluster4` is used with virtual primaries, the order in which the sequences are included affects the computation differently. When longer sequences are included first, although more comparisons are performed, there are fewer cluster primaries to be compared against. Similarly, less computation is spent on updating the virtual primaries, as more of the sequence is provided within the longest sequences. The order of inclusion effect was tested using a dataset of 11,058 sequences with an average length of 460bp, `UIcluster4` was run while including the sequences in two different orders. Adding the sequences in order of descending length, resulted in a clustering run-time of 169 seconds and generated 7485 clusters. In comparison, when adding the sequences in ascending order of sequence length the clustering run-time was 193 seconds and generated 7571 clusters.

5 Results

5.1 Description of Experiment

To evaluate the performance of different clustering methods, several data sets from *Arabidopsis thaliana* and *Homo sapiens* were used. The methods compared

include parallelizing on the cluster space (`UIcluster3`), parallelizing on the input space (`UIcluster4`), and the suffix-tree based method of `PaCE`. The `PaCE` clustering program [8] was included to analyze both parallel speedup and memory requirements. The publically available UniGene clusters from NCBI were used to asses the accuracy of the results. The system used in this comparison was a 16 node, dual processor cluster of 500MHz Pentium III's, each equipped with one Gigabyte of memory. The human EST data set consisted of 41,197 sequences with an average length of 403 bp. The *Arabidopsis thaliana* EST data set contained 81,414 sequences with an average length of 411 bp. The latter data set was used for comparing the accuracy between the programs. The use of *A. thaliana* rather than human ESTs was important in reducing the effect of known genes on the purely sequence-based clustering.

5.2 Accuracy Assessment

Although performance is critical in making the clustering results available, they must also provide an accurate reflection of the underlying mRNA transcripts from which the cDNAs were derived. To assess the accuracy of the clustering methods, two separate comparisons were performed. Both used sets of *Arabidopsis thaliana* sequences. The first data set compared the clustering of 81,414 *A. thaliana* ESTs. This set had previously been clustered with `PaCE` by Srinivas Aluru from Iowa State University. The resulting clusters between `UIcluster4` and `PaCE` were very similar, with 23,642 clusters and 23,995 clusters identified respectively.

A second assessement of clustering accuracy was performed using the complete set of *A. thaliana* ESTs and mRNAs from GenBank. In this assessment, `UIcluster4` was compared to NCBI's UniGene build for *A. thaliana*. The UniGene build contained a total of 27,248 clusters including 9,191 singletons. Similar results were produced with `UIcluster4`, identifying 23,925 clusters of which 6,682 were singletons. This result indicates that `UIcluster4` is more aggressive in merging sequences into the same cluster, resulting in a more conservative estimate of clusters numbers.

5.3 Performance Assessment

Both memory utilization and computation time were measured across these data sets. Table 1 presents the execution time for the same analyses. In this comparison, `UIcluster3` requires approximately one-tenth of the time of `PaCE`. As the number of input sequences increased, the relative difference in computation time between `UIcluster4` and `UIcluster3` decreased. With only 5000 sequences, `UIcluster4` required approximately 60% longer than `UIcluster3` on the same set of sequences. However, on the set of 30,000 sequences, that difference was only 26%. A similar reduction in computation time is observed between `PaCE` and `UIcluster3` with `PaCE` requiring approximately 16 times more computation for 5000 sequences, but only 12 times more in the data set of 20,000 sequences.

The peak memory utilization was assessed on a single node with 1GB of memory, using a subset of the human EST data set. Figure 2 shows the peak memory usage by the three clustering programs. Values were unavailable for `PaCE` with the 30,000 EST data set, as it exhausted the available memory. Note from this figure that the memory requirements of `UIcluster4` increase faster than `UIcluster3` as the number of input sequences grows. This is expected, because the likelihood of novel subsequences that must be included in the virtual primary increases as the number of sequences within a cluster increases, Although the `PaCE` program has to use at least two nodes (one master and one slave node) only the memory utilization for the slave node was measured, because it performs sequence comparisons. If the same computation is run in parallel with `UIcluster4`, the memory requirement per node is significantly reduced, as there are fewer clusters to be stored.

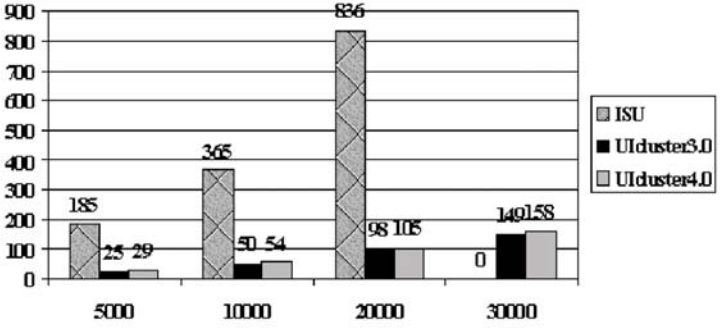

Fig. 2. Memory Utilization

Table 1. Execution time performance comparison.

Num of sequences	*PaCE*	*UIcluster3*	*UIcluster4*
5,000	10 min	37 sec	1 min
10,000	28 min	2 min 7 sec	3 min 28 sec
20,000	1 hr 44 min	8 min 42 sec	12 min
30,000	Out of Mem	20 min	25 min 12 sec

A final performance analysis was performed using the complete set of human EST and mRNA sequences from the human UniGene build. The parallelization on the input space method was used to predict the final number of clusters and the computation time required. This data set contained nearly 4.2 million sequences. The clustering, utilizing 12 nodes, requires an estimated computa-

tion time of 100 hours. For this experiment, the data set was divided into 12 files each containing one twelfth of the ESTs. Thus each file contained roughly 400,000 EST sequences. All of the sequences longer than 1,100 bp were put into a separate file. These thirteen sequence files were first clustered individually. The resulting cluster files were then clustered together to compute the complete set of clusters for the 4.2 million EST sequences. Unfortunately, the final clustering step required more memory that was available. Therefore, the computation time of that component was estimated.

6 Conclusions

An alternative scheme for parallel clustering using `UIcluster` has been described in this paper. The concept of a representative sequence made from the non-redundant set of subsequences from a cluster's constituent sequences is also presented. Such representative sequences can provide further information to biologists regarding several features of biological interest that might otherwise be overlooked. The program is comparable in accuracy to other clustering programs, but requires less computation time. Depending upon the nature of the data set, either of the parallelization schemes may be used to optimize the memory or computation requirements.

Acknowledgements. The authors would like to thank Dr. Volker Brendel from Iowa State University for providing us with the test set of 81,141 *A. thaliana* ESTs, Dr. Srinivas Aluru and Anantharaman Kalyanaraman from Iowa State University for their assistance in obtaining and using the `PaCE` clustering program, and Thomas Bair, Dylan Tack, Jason Grundstad, Jared Bischof, Brian O'Leary and Jesse Walters for their help and suggestions.

References

1. Boguski, M.S., Lowe, T.M., Tolstoshev, C.M.: dbEST – database for 'expressed sequence tags'. Nature Genetics **4** (1993) 332–333
2. Hillier, L., Clark, N., Dubuque, T., Elliston, K., Hawkins, M., Holman, M., Hultman, M., Kucaba, T., Le, M., Lennon, G., Marra, M., Parsons, J., Rifkin, L., Rohlfing, T., Soares, M., Tan, F., Trevaskis, E., Waterston, R., Williamson, A., Wohldmann, P., Wilson, R.: Generation and analysis of 280,000 human expressed sequence tags. Genome Research **6** (1996) 807–828
3. Schuler, G.D.: Pieces of the puzzle: expressed sequence tags and the catalog of human genes. Journal of Molecular Medicine **75** (1997) 694–698
4. Bonaldo, M.F., Lennon, G., Soares, M.B.: Normalization and subtraction: two approaches to facilitate gene discovery. Genome Research **6** (1996) 791–806
5. http://www.ncbi.nlm.nih.gov/UniGene/build.shtml
6. Adams, M.D., Kerlavage, A.R., Flieshmann, R.D., Fuldner, R.A, Bult, C.J., Lee, N.H., Kirkness, E.F., Weinstock, K.G., Gocayne, J.D., White, O.: Initial assessment of human gene diversity and expression patterns based upon 83 million nucleotides of cDNA sequence. Nature **377** (1995) 3–17

7. Miller, R.T., Christoffels, A.G., Gopalakrishnan, C., Burke, J.A., Ptitsyn, A.A., Broveak, T.R., Hide, W.A.: A comprehensive approach to clustering of expressed human gene sequence: The Sequence Tag Alignment and Consensus Knowledgebase. Genome Research **9** (1999) 1143–1155
8. Kalyanaraman, A., Aluru, S., Kothari, S.: Space and time efficient parallel algorithms and software for EST clustering. International Conference on Parallel Processing (2002) 331
9. Trivedi, N., Bischof, J., Davis, S., Pedretti, K., Scheetz, T.E., Braun, T.A., Roberts, C.A., Robinson, N.L., Sheffield, V.C., Soares, M.B., Casavant, T.L.: Parallel creation of non-redundant gene indices from partial mRNA transcipt. Future Generation Computer Systems **18** (2002) 863–870
10. Message Passing Interface Form : MPI: A message-passing interface standard. University of Tennessee Technical Report (1994) CS–94230