A Parallel Expressed Sequence Tag (EST) Clustering Program

Kevin Pedretti, Todd Scheetz, Terry Braun, Chad Roberts, Natalie Robinson, and Thomas Casavant

Parallel Processing Laboratory,
and The Coordinated Laboratory for Computational Genomics,
Dept. of Electrical and Computer Engineering University of Iowa,
Iowa City IA 52242, USA

Abstract. This paper describes the UIcluster software tool, which partitions Expressed Sequence Tag (EST) sequences and other genetic sequences into "clusters" based on sequence similarity. Ideally, each cluster will contain sequences that all represent the same gene. If a naïve approach such as an NxN comparison (N is the number of sequences input) is taken, the problem is only feasible for very small data sets. UIcluster has been developed over the course of four years to solve this problem efficiently and accurately for large data sets consisting of tens or hundreds of thousands of EST sequences. The latest version of the application has been parallelized using the MPI (message passing interface) standard. Both the computation and memory requirements of the program can be distributed among multiple (possibly distributed) UNIX processes.

1 Introduction

Clustering is the process of taking a set of elements and partitioning them into meaningful groups. In the high throughput gene sequencing activities of our laboratories, we generate large numbers of short sequences – Expressed Sequence Tags (ESTs) – and partition them into sets based on similarity. The importance of this problem bears on several aspects, but the principal of these are creating non-redundant indices of genes and assessing the novelty of sequencing. If done in a naïve fashion, such as a NxN comparison, this problem would be intractable for the data set sizes we produce (50K–300K ESTs). Although there are several existing software system [7,5,1,6] available that perform sequence clustering accurately, our program is unique in its ability to efficiently and accurately cluster EST sequences. Over the past four years, we have developed techniques to speed up the computation by using increasingly sophisticated heuristics along with parallel processing techniques. The usefulness of our program, `UIcluster`, has been demonstrated in the identification of more than 100,000 unique/novel clusters across three species (human, mouse, and rat).

2 Expressed Sequence Tags (ESTs)

From a biological perspective, ESTs are partial transcripts of genes. Specifically, they are sequenced from cDNA (complementary DNA) clones, synthesized from polyA-selected whole-cell RNA. To prepare for EST sequencing, mRNA molecules are extracted from cells and converted into cDNA through reverse transcription. The cDNAs are then cloned into a vector and electroporated into bacteria for growth, amplification, and storage. A collection of such cDNAs is referred to as a library. Each cDNA library potentially contains many unique and previously undiscovered genes. However, significant redundancy within a library (multiple copies of the same mRNA) and between libraries is normal.

High throughput EST sequencing for gene identification involves sequencing the 3' end of randomly chosen cDNA clones from a cDNA library. The use of a poly-T primer during reverse transcription allows for the preferential creation of cDNAs with a poly-A tail at their 3' ends. Thus, sequencing can start from a known position (within poly-A tail).

For the purposes of this paper, and from the computational perspective, an EST is a character string made up of letters from the alphabet A, C, T, G, X, N where A, C, T, and G represent the four nucleotide bases of DNA and X and N represent bases within repetitive (low-complexity) segments or that are of indeterminate identity. ESTs are typically between 400 to 1000 letters, or bases, long. Comparing pairs of ESTs and looking for similarity is the basic element of clustering. This comparison is complex because the underlying sequencing technology is error prone – bases can be inserted, deleted, or misread. Studies of our EST sequences have indicated that the error rate for EST sequencing is approximately 5% for misread errors, and 1-2% for insertion/deletion errors.

3 Uses of Clustering

Clustering is used to assess the gene discovery rate of sequencing done from cDNA libraries. For single library assessment, the entire set of ESTs obtained from that library is used as a input for clustering. Clustering partitions the set into subsets, or clusters, based on similarity. Each EST is a member of at most one cluster. Novelty is computed as the number of clusters identified divided by the number of sequences clustered.

This computation is used to calculate both incremental and overall novelty rates (roughly corresponding to gene discovery rates) for individual cDNA libraries and for EST projects as a whole. Incremental novelty calculations are performed daily to monitor the sequencing efforts and to determine when cDNA library subtractions should occur [2]. This procedure can dramatically increase novelty rates. However, the subtraction process is time consuming and cannot be performed on a continual basis.

Figure 1 shows an example of the effectiveness of these procedures for a progression of four cDNA libraries, named C0, C1, C2p, and C3. Each sharp increase in novelty rates corresponds to a subtraction on the preceding library being performed.

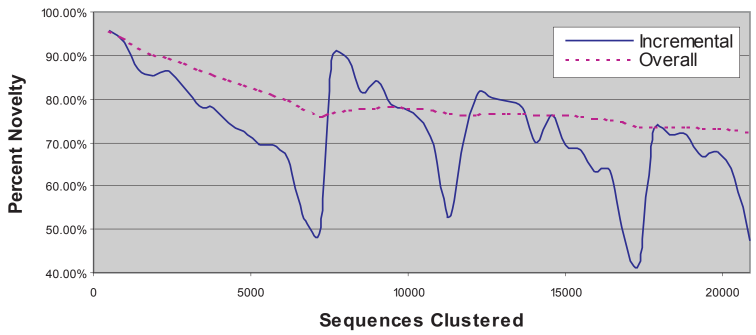

Fig. 1. Incremental library novelty

Another significant use of clustering is the generation of non-redundant gene indices, or UniGene sets [7]. As mentioned previously, ideally each cluster will uniquely represent a gene. Thus, the goal in constructing a UniGene set is to bring together all of the ESTs sequenced for a given gene into a single cluster. This information is useful for reducing redundant processing and for the annotation of EST sequences.

4 Program Evolution

`UIcluster` has evolved as our laboratory's processing requirements have increased. Three generations of the clustering program have been developed to date. The first revision was developed to work well for moderately sized data sets of ESTs. As our data sets grew, this version required more than a days computation time to cluster the entire set of ESTs. The main goal of the second version of the program was improved performance for large data sets. A third, parallelized version provided higher performance and several additional features has recently been released. All revisions of `UIcluster` may be freely obtained from our project web site (http://genome.uiowa.edu).

The basic clustering program flow proceeds as follows: 1) read one sequence from the input file, 2) compare the sequence against every existing cluster, 3) based on sequence similarity, either add it to an existing cluster or make it the first member of a new cluster. This process is repeated until every sequence in the input file is examined. In step 3, the EST is only added to an existing cluster if the specified similarity criteria is met. The similarity criteria is run-time configurable and is of the form N out of M bases. For example, 38 out of 40 bases would mean two sequences are judged to be similar if there is at least one window of 38 out of 40 bases in common, allowing insertion, deletion, and mismatch errors. The speed of the program is directly effected by these parameters. Higher error tolerance $(M - N)$ increases program execution time significantly as does larger window sizes (M).

```
Sequence: GCCACTTGGCGTTTTG
Hashes:
          Hash 1: GCCACTTG = 48406
          Hash 2: CCACTTGG = 44869
          Hash 3: CACTTGGC = 27601
          Hash 4: ACTTGGCG = 39668
          Hash 5: CTTGGCGT = 59069
          ...etc.
```

Fig. 2. Example of hashing a sequence

4.1 Revision 1.0

Revision 1.0 was useful for relatively small data sets ($<$ 30,000 ESTs). The program was structured so that clusters were stored in a 2-D linked list. Each EST read from the input file was compared against a single representative element from each cluster. The longest EST from each cluster was used as a representative element for that cluster.

Evaluating the N of M similarity criteria for two sequences is computationally intensive. As a performance optimization, we used a hashing technique to eliminate comparisons that will obviously be unsuccessful (i.e., the N of M criteria will not be met). A *hash* is simply an integer that uniquely represents a short string of characters. The general equation used to generate a hash is given by (1).

$$H = \sum_{i=0}^{\zeta-1}(K^i * \phi_i) \tag{1}$$

In this equation, H is the generated hash value, ζ is the string length, K is the alphabet size, and ϕ_i is the integer value assigned to the letter at position i in the string being hashed. The string length ζ that can be used to generate hashes is limited by the word size of the computer. For the DNA alphabet, each base requires 2-bits to represent it ($\lceil \log_2 K \rceil$ where $K = 4$ for DNA). Thus, the maximum value of ζ using a single word on a 32-bit machine is 16.

When a sequence is hashed, equation 1 is used on every ζ length sub-string. Figure 2 shows the first six hashes generated for a sample sequence with $\zeta = 8$.

When an EST is clustered, the N of M similarity criteria is only evaluated for cluster representatives that contain one or more hashes in common with the EST being clustered. The length of the hash probe used is an important parameter that can significantly affect performance. Longer hash lengths will result in better performance for a given similarity criteria. It must also be chosen carefully so that potential similarities are not missed. The formula for calculating the maximum hash size is shown in (2). The rational for this equation is that for any chosen similarity criteria N of M, there is at least one contiguous, error-free region of ζ bases. Thus, the comparison of two sequences can be accelerated by first searching for short exact matches of length ζ bases between the pair (i.e. searching for identical hashes). If such a match is found, a more exhaustive

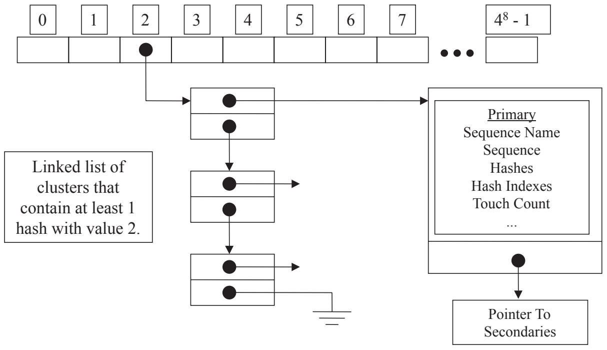

Fig. 3. Global hash table

search that permits errors can be performed. If no length ζ hashes are identified, then the two sequences cannot possibly contain a window of M bases with N bases in common.

$$\zeta = \left\lfloor \frac{M}{M - N + 1} \right\rfloor \tag{2}$$

The calculation to generate the hashes for a sequence is only performed once since the hash lists are stored in memory. However, the hashes are accessed many times during the programs execution. This amortizes the computational overhead of generating the hashes.

4.2 Revision 2.0

The main improvement in revision 2.0 was the implementation of the global hash table (GHT). As our EST data sets grew larger, the sequential nature of the traversal of the cluster representative linked list for every input sequence became a bottleneck. The GHT optimizes the program at a higher level than individual sequence comparisons by filtering the entire search space of cluster representatives into a subset of high-potential candidate targets.

When a new sequence is clustered, a list of hashes is generated for each ζ base window of its sequence. Each hash in the list is then used as an index into the GHT. Figure 3 shows a GHT with 4^8 elements, corresponding to $\zeta = 8$. Each element in the table points to a linked list of clusters that contain at least one occurrence of the hash equal to its index. In figure 3, there are three clusters that contain the hash 2. If the sequence being clustered also has a hash of two, the touch count field of each cluster linked from the second element in the GHT is incremented. If the touch count field of a cluster exceeds a run-time configurable threshold, a detailed sequence comparison is performed between the input sequence and the candidate cluster. This procedure is based on the premise that two similar sequences will likely have many hashes in common.

Care must be taken to adjust the touch count threshold appropriately. For a given similarity criteria (e.g. 38 out of 40 bases) and hash length ζ, if the

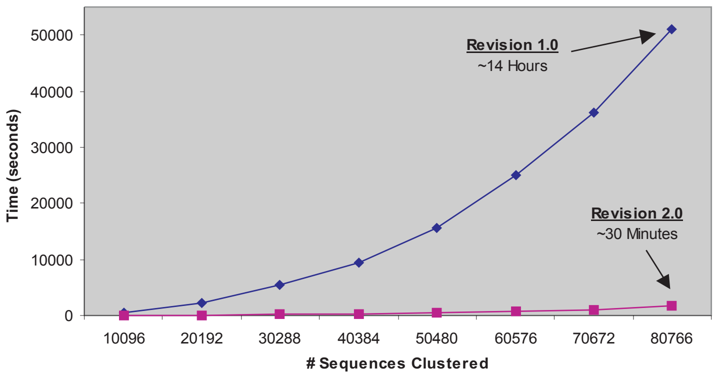

Fig. 4. Execution time of Revision 1.0 vs. Revision 2.0

threshold is too low the speedup due to the GHT will be small. Conversely if the threshold is too high, some sequence similarities will be missed.

Figure 4 shows the execution time for both revisions of the clustering program with an input data set of 80,766 rat EST sequences. Revision 2.0 demonstrates 28x speedup while calculating virtually identical results. The major trade-off of the GHT optimization is memory utilization. However, on a 2GB machine we have been able to cluster data sets as large as 1 million ESTs. While theoretically the first revision could handle data sets this long, the computation time required would make it impractical.

4.3 Revision 3.0

The latest version of the clustering program has been parallelized to split up the computational and memory requirements across several computers (compute nodes). The main reasons for doing this are for added performance and so that the program can scale to larger problem sizes without being constrained by the memory limitations of a single computer. The MPI (message passing interface) [4] communication standard has been used for inter-process communication.

In this mode of execution, each cluster is stored on exactly one compute node. A given sequence is read in from the input file and processed in parallel on each compute node. This results in a parallel search of the cluster space. Once each node has finished its search, each node's best match is collectively communicated to all compute nodes. The node with the best match stores the sequence in its memory space. If no match is found on any of the compute nodes, the input sequence becomes a new cluster and is assigned to one of the compute nodes. Clusters are balanced evenly across the compute nodes.

Figure 5 illustrates the parallel speedup obtained for clustering a data set of approximately 81,000 rat EST sequences. The three curves represent three different runs of the program using different parameter sets. The first curve (labeled 1) corresponds to the default parameters used in our processing pipeline.

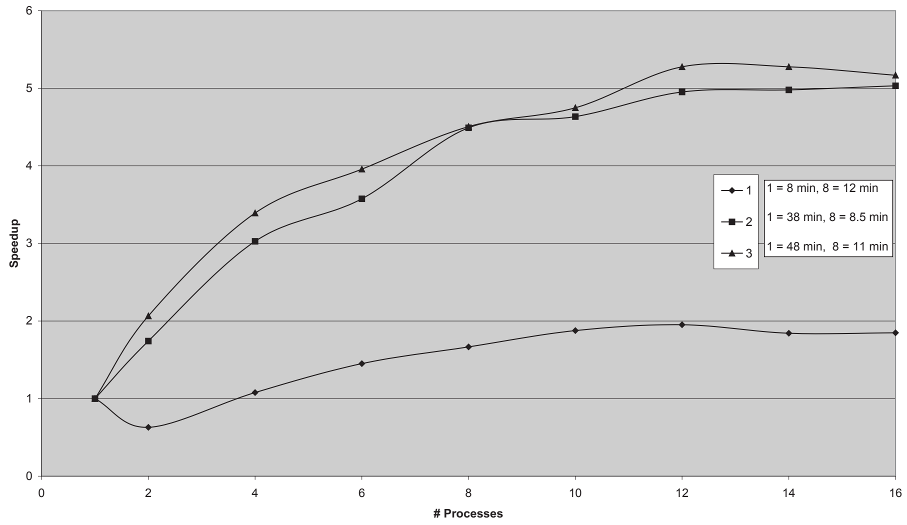

Fig. 5. Parallel speedup

The second curve (labeled 2) adds the extended search option. By default, an EST is added to the first cluster it found to be similar to and the search is halted. This option enables the identification of all similar cluster representatives for each EST clustered. The third curve (labeled 3) enables the reverse complement checking option of the program.

Since the implementation uses a collective communication at the end of every sequence clustered, the amount of computation required for each sequence is important. As the grain size increases, better performance should be observed since relatively less communication is being performed.

The times in minutes for the single and 8 node run for each case are shown in the figure. Performance scales poorly for the first case, actually decreasing when using two compute nodes. This is most likely due to the computation being unevenly distributed and the communication overhead. With more compute nodes, performance increases somewhat but is never greater than double that of the serial case. The larger grain size of the second case results in significantly improved speedup. The third curve scales similarly since the grain size is only slightly increased for this case.

5 Conclusion

The evolution of an EST clustering program has been discussed in this extended abstract. Background information on the problem has been presented along with details of two sequential implementations and a parallel implementation. Planned extensions to `UIcluster` include utilizing the recently released human genome sequence [3,8] to improve the accuracy of clustering, and to aid in identification of alternative splice forms and intron/exon boundaries. Other extensions planned include improved performance for long sequences (e.g., full length cDNA sequences), automatic cluster merging, and tools for manual curation of clustering results by expert human operators.

References

1. Adams M.D., Kerlavage A.R., Fleishmann R.D., Fuldner R.A., Bult C.J., Lee N.H., Kirkness E.F., Weinstock K.G., Gocayne J.D., White O., et al. (1995) Initial assessment of human gene diversity and expression patterns based upon 83 million nucleotides of cDNA sequence. Nature 377:3-17
2. Bonaldo M.F., Lennon G., Soares M.B. (1996) Normalization and subtraction: two approaches to facilitate gene discovery. Genome Research 6:791-806
3. International Human Genome Sequencing Consortium (2001) Initial sequencing and analysis of the human genome. Nature 409:860-921
4. Message Passing Interface Form (1994) MPI: A message-passing interface standard. University of Tennessee Technical Report CS-94-230
5. Miller R.T., Christoffels A.G., Gopalakrishnan C., Burke J.A., Ptitsyn A.A., Broveak T.R., Hide W.A. (1999) A comprehensive approach to clustering of expressed human gene sequence: The Sequence Tag Alighment and Consensus Knowledgebase. Genome Research 9:1143-1155
6. Parsons J.D., Brenner S., Bishop M.J. (1992) Clustering cDNA Sequences. Computational Applications in Bioscience 8:461-466
7. Schuler G.D. (1997) Pieces of the puzzle: expressed sequence tags and the catalog of human genes. Journal of Molecular Medicine 75:694-698
8. Venter J.C., Adams M.D., Myers E.W., Li P.W., Mural R.J., Sutton G.G., et al. (2001) The sequence of the human genome. Science 291:1304-1351