BIOINFORMATICS APPLICATIONS NOTE Vol. 19 no. 5 2003, pages 651–652
DOI: 10.1093/bioinformatics/btg034

TIGR Gene Indices clustering tools (TGICL): a software system for fast clustering of large EST datasets

Geo Pertea [1,*], Xiaoqiu Huang [2], Feng Liang [1,3], Valentin Antonescu [1], Razvan Sultana [1], Svetlana Karamycheva [1], Yuandan Lee [1], Joseph White [1], Foo Cheung [1], Babak Parvizi [1], Jennifer Tsai [1] and John Quackenbush [1,*]

[1]The Institute for Genomic Research, Rockville, MD 20850 USA, [2]Department of Computer Science, Iowa State University, Ames, IA 50011 USA and [3]Current address Invitrogen Corporation, Carlsbad, CA 92008 USA

Received on July 10, 2002; revised on August 12, 2002; accepted on August 16, 2002

ABSTRACT

TGICL is a pipeline for analysis of large Expressed Sequence Tags (EST) and mRNA databases in which the sequences are first clustered based on pairwise sequence similarity, and then assembled by individual clusters (optionally with quality values) to produce longer, more complete consensus sequences. The system can run on multi-CPU architectures including SMP and PVM.

Availability: http://www.tigr.org/tdb/tgi/software/

Contact: johnq@tigr.org; gpertea@tigr.org

Expressed Sequence Tags (ESTs) have provided insight into transcribed genes in a variety of organisms and are widely used for gene discovery and expression analysis. However, identifying encoded genes from ESTs presents a number of challenges. ESTs represent large numbers of redundant, typically partial, transcript sequences from diverse biological sources, low quality sequences often without quality scores, relatively frequent chimaerism, and a moderate rate of vector and adapter contamination. Most sequence assembly programs developed for genomic applications are not well adapted for ESTs. Further, the distribution of ESTs can conspire to produce incorrect and chimaeric assemblies (Liang *et al.*, 2000). Careful partitioning of sequences prior to assembly is essential for faithful transcript reconstruction. TGICL uses stringent pairwise comparisons between input sequences to group those sharing significant regions of near identity. Individual assembly of each cluster has the advantage of producing larger, more complete consensus sequences while eliminating potentially misclustered sequences.

In its simplest application, TGICL takes a single parameter: an input multi-FASTA file. TGICL's final output is one or more ACE files containing CAP3 assemblies (Huang and Madan, 1999) and a list of singletons. Prior to running TGICL, the input dataset should be cleaned to remove contaminating sequences, including vector, adapter, and bacterial sequences, which can lead to misclustering and misassembly. This can be done using either a stringent program such as Lucy (Chou and Holmes, 2001) or a sequence trimming script such as SeqClean (http://www.tigr.org/tdb/tgi/software) with filtering databases such as NCBI's UniVec. Known repeats should also be masked using RepeatMasker (Smit and Green, unpublished) with the lower-case masking option; TGICL excludes masked regions during its initial word-hashing phase.

Known full-length transcripts can be used for 'seeded clustering', which helps create smaller, better partitioned clusters and avoids chimeric assemblies. Seeded clustering assumes that a complete gene transcript has nearly perfect identity with all ESTs from that gene; lateral extension of seeds is limited to nearly perfect alignments. Seeded clustering uses a reserved prefix for full-length transcript names in the input FASTA database (i.e. 'et|') and takes as a parameter the minimum number of sequences in a single-linkage cluster that define 'seeded re-clustering'. One should be cautious in using seeds, as partial sequences mislabeled as complete can prevent cluster extension beyond the seed.

Clustering first indexes the input file and then performs all-versus-all pairwise similarity searches using mgblast, a modified version of megablast (Zhang *et al.*, 2000). mgblast differs from original in that it allows specific output filtering options including minimum overlap length and identity, uses a dynamic offset within the database when performing incremental searches, and produces a tab-delimited output with one line for each identified overlap.

*To whom correspondence should be addressed.

Downloaded from https://academic.oup.com/bioinformatics/article/19/5/651/239299 by guest on 23 April 2024

Large databases can be partitioned into separate files (slices) that are searched against the entire database. This allows for parallel processing as multiple slices can be searched with independent mgblast processes; both SMP and PVM architectures are supported. Results from each search are sorted and compressed after all slices are processed. Results are merged and sorted into incrementally numbered, compressed files. The criteria for initial sorting and subsequent merges is decreasing pairwise alignment score. This global sorting uses a 'greedy' approach to controlled clustering: the best alignments are encountered first and both initiate and direct cluster formation. When seeded clustering is used, overlaps with full-length mRNAs are processed first and constitute the cluster 'seed'. Sequence overlaps in the sorted results files are filtered using user-defined criteria: the minimum length of the overlap (default 40 basepairs), the minimum percent of identity for the overlap (default 95%), and the maximum mismatched overhang (dynamically adjusted for long sequences and overlaps; the default starts at 30 nucleotides).

Three clustering utilities in TGICL, tclust, sclust, and nrcl, use internal graph representations where sequences represent nodes and filtered alignments represent edges. As overlaps are processed, clusters are formed incrementally through transitive closure, resulting in a set of connected sub-graphs representing clusters. These are written to a file with the largest clusters first; each cluster represented by a list of component sequences. For seeded clustering, overlaps involving known full-length transcripts are assessed differently as the coverage of other sequences in the cluster should be nearly complete. If a seeded cluster does not meet minimum coverage standards, the seed is ignored.

For assembly, a multi-FASTA file is built from each cluster and passed to CAP3 (or a user-specified assembler) for multiple alignment and consensus building. As clusters are independently assembled, this can be executed in parallel using multiple CPUs. Quality values are not required, but if used, quality values for sequences in each cluster are collected simultaneously with the FASTA sequences and submitted to the assembler. CAP3 produces both consensus sequences and ACE files that can be used for further analysis. A list of sequences not falling into any assembly is written as a 'singleton' file. A graphical, interactive ACE file viewer (clview) is available (http://www.tigr.org/tdb/tgi/software/).

TGICL was developed and tested under Linux on single and dual-CPU machines and on a Linux PVM cluster. The main program is a perl script that provides brief instructions when run without parameters. Parallel processing utilities called by the main program, psx and pvmsx, are written in C: psx uses 'fork' call to spawn multiple processes, while pvmsx uses the PVM API to distribute work over independent PVM nodes. The clustering programs, the merge utility, and the FASTA record indexing/retrieval programs are written in C++. Pairwise searches are performed by mgblast, written in C. As source code is provide, all programs in TGICL can be recompiled on other Unix systems. CAP3 is distributed only as executables for Linux and Sun platforms; other binaries can be requested from the program author (Huang and Madan, 1999).

Clustering is very fast due to the modified megablast engine used for pairwise searches and distributed processing makes TGICL even faster: on a PVM cluster with 20 Pentium III nodes, an input file of 1 700 000 entries was clustered in approximately one hour and assembly was completed the following day. Sets of 150 000 sequences can be fully clustered and assembled overnight on a single CPU.

TGICL has difficulty with highly expressed genes that have several thousand ESTs in a single cluster. For these, CAP3 or other assemblers generally run out of memory. TGICL does not yet deal with such clusters automatically, but the command line clustering utilities sclust and nrcl can offer insight into the structure of large clusters and can enable development of customized assembly solutions such as an iterative (two-step) assembly, or the use of containment clustering to determine the representative sequences to create a scaffold on which unassembled sequences are mapped.

TGICL is used to generate the TIGR Gene Indices (TGI; Quackenbush *et al.*, 2001; http://www.tigr.org/tdb/tgi/), representing independent analyses for nearly 60 species with EST collections of fewer than 10 000 to more than 4 000 000 sequences. These databases have demonstrated their utility through annotation of eukaryotic genomes and cDNA collections, identification of orthologous genes, and annotation of microarray resources. TGICL updates will be available as they are developed.

REFERENCES

Quackenbush,J., Cho,J., Lee,D., Liang,F., Holt,I., Karamycheva,S., Parvizi,B., Pertea,G., Sultana,R. and White,J. (2001) The TIGR Gene Indices: analysis of gene transcript sequences in highly sampled eukaryotic species. *Nucleic Acids Res.*, **29**, 159–164.

Liang,F., Holt,I., Pertea,G., Karamycheva,S., Salzberg,S. and Quackenbush,J. (2000) An optimized protocol for analysis of EST sequences. *Nucleic Acids Res.*, **28**, 3657–3665.

Chou,H.-H. and Holmes,M.H. (2001) DNA sequence quality trimming and vector removal. *Bioinformatics*, **17:12**, 1093–1104.

Huang,X. and Madan,A. (1999) CAP3: a DNA sequence assembly program. *Genome Res.*, **9**, 868–877.

Smit,AFA and Green,P. (unpublished) unpublished. See http://ftp.genome.washington.edu/cgi-bin/RepeatMasker.

Zhang,Z., Schwartz,S., Wagner,L. and Miller,W. (2000) A greedy algorithm for aligning DNA sequences. *J Comput Biol.*, **7**, 203–214.

Downloaded from https://academic.oup.com/bioinformatics/article/19/5/651/239299 by guest on 23 April 2024