Rapid and Incremental sequence Clustering and Assembly repository database

What's the need of assemblies?

The sequence assembly problem

The limit of current sequencing technologies is around 500 up to 800 nucleotides. Thus to sequence long EST/mRNA, we need to split the sequence (in fact several copies) into shorter pieces having compatible lengths with the sequencing technology. These shorter sequences are obtained via a random process; either chemical (digestion by an enzyme), or mechanical (sonication).

Once these fragments have been sequenced, we face with a kind of puzzle: how to rearrange these randomly obtained fragments in order to rebuild the initial sequence?

Because of the random process, some of these sequences should overlap, and if enough fragments have been sequenced we should have a full coverage of the initial sequence.

The benefits of assembly

Having longer sequences has several advantages:

- The way they are obtained, i.e. consensus of aligned overlapping sequences, improve the quality of the sequence, removing sequencing errors,
- It allows having a more precise idea about the redundancy inherent in the sequencing of clones. This is particularly useful to decide when to build subtractive libraries,
- It's a way to have full length clones, and thus having their lengths, which can be helpful for instance to choose clones to be spotted on micro arrays,

- The longer is the sequence, the better is the annotation we can make through homology searches.

R-Icard and the Sigenae assembly projects

Two kinds of assembly projects exist:

- Public projects which gather for a given species EST or mRNA sequences available in public sequence databases (respectively dbEST and EMBL). All contigs obtained, including singlets, are kept and are publicly available through the “Agenae EST Contig browser”. Public projects will be updated regularly and new assembly processed on a base of one per month (if, of course, the amount of new sequences is sufficient).
- Private projects which gather the same public sequences and also private sequences coming from AGENEA EST sequencing programs. If a private sequence has been published into a public database, it should appear twice in the assembly project: one time with its private name, the second time with its identifier in the public database. Taking benefits of public sequences in this kind of project allows us to lengthen the obtained contigs. Only contigs that have at least one private member are kept and can be displayed through the “Sigenae EST Contig browser” for registered people. Private projects will be processed on demand, and will rely on the computations made for the public project associated to the same species.

The way these two kinds of projects are handle is quite similar. Among the different software tools available to process the assembly, we chose CAP3[1] because of its low cost (free for academics), its performance, and its capability to take account of the quality of sequenced nucleotides into the computation of the alignment score.

Before the assembly can be started, three prior steps are needed: i) the computation of the qualities, ii) some sequence cleaning to reject those contaminated or of too poor quality, or to remove flanking sequence parts (clipping) which doesn’t belong to the clone insert, and finally iii) a rough clustering of these sequences to split the full

[1] Huang X. and Madan A. (1999) *CAP3: a DNA sequence assembly program.* Genome Research, 9:868-877

set into smaller enough ones to be processed by CAP3, whose limit is around 100.000 sequences (less more than the amount of sequences to be processed now).

Once the assembly is made and contigs obtained, these latter are annotated through homology searches against several databases (nucleic or proteic).

Quality computation and sequence cleaning

The two first steps are included into the SURF workflow, and resulting data are stored into SURF databases.

To compute the qualities of private sequences, SURF uses Phred[2][3] with default parameters, but with ad hoc chemistries/dyes definitions, to process each raw chromatogram and generate associated FASTA sequence and quality files (phred scores computed for each base read). For public sequences, for which chromatograms are not available, pseudo qualities are generated to mimic the quality profile of EST or mRNA.

Vector/adapter detection is performed with cross_match[4] software against clone library specific vector/adapter definition files (when available, Univec otherwise). Some other features are also detected, like for instance polyA/T or repeats (respectively with home made software and RepeatMasker[5] against species specific repeats database extracted from RepBase[6]).

Based on these features and the quality of the sequences, the boundaries of clone inserts are computed by SURF and then contamination checking is performed against several sequence databases: Univec, Yeast and E. coli genomes as well as species specific ribosomal and mitochondrial genomes (complete or partial version, depending on the species). Finally, a status is computed for each sequence denoting if the sequence is valid or not for an assembly, i.e. no contamination and enough informative nucleotides.

---

[2] Ewing B., Hillier L., Wendl M., and Green P. (1998) *Basecalling of automated sequencer traces using phred. I. Accuracy assessment.* Genome Research 8:175-185.
[3] Ewing B. and Green P. (1998) *Basecalling of automated sequencer traces using phred. II. Error probabilities,* Genome Research 8:186-194
[4] Green P.(1993-1996) http://www.phrap.org/phredphrap/general.html
[5] Smit, AFA & Green, P RepeatMasker at http://repeatmasker.org
[6] Jurka, J.Repbase Update (2000): *a database and an electronic journal of repetitive elements*. Trends Genet. 9:418-420

Sequences that will be extracted from SURF databases are those having a valid status, and the portion used will be restricted to the insert part.

It is very important for the quality of the assembly to have an accurate cleaning of the sequences, this to avoid chimerical contigs linking sequences by their polyA trail or unclipped vector for instance.

Rough clustering of sequences

Because of the limitation of CAP3 in terms of number of sequences it can handle, and also to speed up the computations, a rough clustering of sequences to be assembled is needed.

The purpose is to split the whole set of sequences into several smaller ones. Of course, we should put into the same subset (we call it cluster) sequences that could participate to a same contig. The best would be to group overlapping sequences in the same way as CAP3.

To summarize how CAP3 process with default parameters[7], it first clips the sequences based on their qualities, then it looks for overlaps that mainly fulfill i) a minimum length (-o = 30) conserved at a given percent of identity (-p = 75), and ii) a minimal alignment score (-s = 500) taking into account the base qualities.

The challenge here is to gather into the same cluster sequences that share segments of 30 nucleotides conserved at 75% of identity with a minimal alignment score.

However, the rough clustering i) will not be processed on CAP3 like quality clipped sequences, but on full insert, and ii) it will use fast sequence comparison program that don't take into account the quality for the computation of alignment scores. Thus, we decided to strengthen the minimal overlap length constraint to 75 nucleotides conserved at 96% of identity.

The software chosen to process the sequence comparison is megablast[8] which performs quick searches for highly similar sequences. To be compatible with the minimal overlap length constraint, we set the –W parameter to 18 (length of best

[7] it's the configuration we chose. More details about CAP3 are available here: http://www.msi.umn.edu/software/cap3/doc

[8] Zheng Zhang, Scott Schwartz, Lukas Wagner, and Webb Miller (2000), *A greedy algorithm for aligning DNA sequences*, J Comput Biol 2000; 7(1-2):203-14.

perfect match). We also modify the following parameters: -X 8 (dropoff value for gapped alignment), -e 1e-3 (Expectation value), and -F to mask into the query sequence dust and species specific repeats (it required some adaptations of megablast).

The output of megablast is then filtered by home made parsers to extract couples of sequences that share a segment of at least 75 bp conserved at 96% of identity.

At this step, we have a list of relations between two sequences that share a minimal desired segment. It can also be seen as a graph definition, where nodes are sequences and arcs between nodes denote that sequences should belong to the same cluster. We call these arcs “related arcs”.

We could extract directly from this graph definition the connected components (the clusters we are looking for). However, for many sequences, even public ones, we have often clone information. Sequences coming from the same clone should belong to the same cluster. We can define a new kind of arc between two sequences: “clone arcs” which denote that sequences are coming from the same clone and should belong to the same cluster.

To be sure that all the sequences extracted will participate to the clustering, we also add arcs between each sequence and itself. We call these arcs “singlets arcs”.

The input graph used to compute the clusters will gather related, clone and singlets arcs.

Obtained clusters can be just singlets or gather sequences that share hoc segment or belong to the same clone.

Splitting of huge clusters

For some species, the redundancy of sequenced EST is very huge. In such a case, some of the clusters will still gather a too large amount of sequences to be computed by CAP3. The only solution to split this kind of cluster is to discard some arcs (only related and clone arcs are concerned here). To do so, several software packages for partitioning graphs exist. Among them, we chose Metis[9] for its simplicity, and mostly

[9] G. Karypis, and V. Kumar (1995) *A fast and highly quality multilevel scheme for partitioning irregular graphs*. SIAM Journal on Scientific Computing.

its performance (the graphs we have to split have usually about 100.000 nodes and more than 3.000.000 arcs). The objective of this traditional graph partitioning problem is to compute a balanced k-way partitioning such that the number of edges (or in the case of weighted graphs the sum of their weights) that straddle different partitions is minimized.

To avoid discarding strong relations between sequences, we add weights to the graph as follows:

- Related arcs are weighted by the highest identity it can exists between two sequences (max of the number of conserved aligned nucleotides in HSPs between two sequences),
- Clone arcs are weighted according to the length of the clone name (length x 15). In public databases mainly, we often have noisy clone names (like “5’”). We decided that the longer is the clone name, the higher is its pertinence.

Clusters will be split as less as possible, to obtain a minimal number of sub clusters that can be processed by CAP3.

Assembly

This step is quite simple. For each cluster obtained having more than one sequence, we extract FASTA sequences and related quality files, which will be given as input to CAP3 with the default parameters. To know more about the way the assembly is performed by CAP3, please refer to (7).

What will be called contigs and stored into the EST Contig Browser are singlets obtained through the clustering step, singlets generated by CAP3, and of course CAP3 contigs.

Naming conventions for contigs

Contigs gather sequences extracted from SURF database. Each sequence has an insertion date attribute in this database. A good way to name a contig is to use the smallest (lexicographic order) name of the oldest sequences it gathers. Like that, even if new sequences are added, the contig should keep this name; expect if the contig is split or merged.

To differentiate contigs from different projects, we add to the contig name a suffix: .p.sp.v, where p is a letter which denotes the project (p for public, s for sigenae, …), sp denotes the species (bt for bos Taurus, sc for sus scrofa, …), and v for the version of the assembly.

In SURF database, a public sequence has a unique name which is its identifier in its source database (dbEST ID, EMBL ID, …), and to which a version number is attached. This name is perfect to identify in an unambiguous way the sequence, but biologists prefer accession numbers. Nevertheless, several sequences can share the same accession number. Thus, for public sequences accession numbers will be used instead of sequence name, and if several sequences share the same accession number, a letter (b...z) will be appended at the end.

A contig name will look like:

- $\underbrace{\text{BP108491}}_{a}\underbrace{.1}_{b}\underbrace{\text{b}}_{c}\underbrace{\text{.p.bt.1}}_{d}$ for contig name based on public sequence, where a) is the accession number of the oldest sequence having the smallest name, b) is its version in SURF database, c) is an optional letter as explained above, and d) is the contig name suffix;
- $\underbrace{\text{bcbs0001.c.12.b\_5.1}}_{a}\underbrace{\text{s.bt.3}}_{b}$ for contig name based on private sequence, where a) is the sequence name, and b) the contig name suffix.

Automatic annotation

Making the annotation of a sequence is almost reduced to homology searches against different kind of databases. For each contig, we use ad hoc filtered NCBI blast programs against the following databases:

- Tigr species specific assembly, except for rabbit
- Unigene species specific assembly, except for rabbit
- Unigene Human
- Current contigs
- Previous version of assembly (if available)
- Public contigs (if available)

- Swissprot
- Prodom
- Human transcripts
- Chicken transcript for chicken assembly (not yet used)
- Zebrafish, Fugu, and Tetraodon transcripts for trout
- Trembl (not yet processed)

We process blast with an expectation value of 1e-5, and keep the 300 first best HSP.The blast parser then tries to filter HSP.

Each HSP defines a segment of homology on the query sequence (the contig). HSP are processed one by one by decreasing score, and if the associated segment of homology on the contig has not been yet marked as such, or if it extends an existing one, the HSP is kept and the segment added or extended. Otherwise, if the HSP defines a segment embedded in a previous one, the HSP is discarded, except for HSP related to the species; in this case all HSP are kept.

For homology searches on genomic sequences the algorithm above is modified as follows: all HSP are kept in such a way to obtain the longest transcript (no missing exons).

Incremental clustering

Clustering is one of the most consuming time steps in an assembly project. Let be N the number of sequences to be assembled, the clustering needs n^2 sequence comparisons.

If we add p sequences, and want to process a new assembly taking into account these new sequences, the clustering step would need $(n+p)^2$ sequence comparisons.

However, sequence comparisons are just needed to produce related arcs. If we keep for each version of assembly the related acs, we just need to compute missing arcs for the next version. Instead of making $(n+p)^2$ sequence comparisons, we just have to make $2np + p^2$ sequence comparisons. This could be reduced to $np + p^2$ sequence comparisons if we won't have used the filtering option of megablast; in our case, comparing n sequences to the database of p sequences is not symmetrical.

To summarize, incremental clustering consists in the reuse of related arcs computed in the previous assembly. Instead of processing a megablast of (n+p) sequences against a database of (n+p) sequences, we process 3 megablast:

- n sequences against a database of p sequences
- p sequences against a database of n sequences
- p sequences against a database of p sequences

Because of p is normally very smaller compared to n, the cost of these computations is significantly reduced.

The only precaution we have to take before processing these megablast, is to add/remove sequences and associated arcs that have their status changed to valid/not valid.

If we consider private assembly projects, that take benefits of public sequences, they can be seen to a certain extent as incremental version of a public project, and thus we can apply the same techniques to reuse related arcs of the public project.

Based on these results of homology search, and taking into account associated knowledge if available (annotation of a protein, gene annotation on a genomic region, …), keywords, GO and a trial of functional annotation are associated to the contig.

Differential Assembly / Annotation

Now, if we compare clusters or contigs between two assembly versions, if a cluster/contig has not changed in terms of its contents (sequences involved in it), the results of computation made on it should be the same.

Therefore, between to versions of assembly we will compare:

1. the clusters obtained, and only make the assembly of changed or new clusters. Results for unchanged clusters can be simply copied from previous version of assembly,
2. the contigs obtained, and only annotate changed or new ones. Results for unchanged contigs can be simply copied from previous version of assembly, except if an update of the database used by the annotation has occurred in between the two versions.

Hardware architecture

The clustering and the assembly are now processed on a cluster of Linux machines. Splitting the computations allows us to process a full assembly project of 300 000 sequences in two or three days.