Concept: How CD-HIT works - Next Genetics

blog.nextgenetics.net/

CDHIT is a program commonly used to cluster nucleotide/protein sequences. It is used routinely by NCBI to get rid of redundant sequences in the NR (non-redundant) database. It is extremely fast compared to a traditional all vs all blast and subsequent pair-wise clustering.

I am going to attempt to explain the algorithm behind CDHIT and the associated advantages and disadvantages.

Algorithm

The algorithm is a greedy incremental clustering algorithm. Basically it means it will try to cluster as much as it can and it does so in some kind of an order. The ordering used in CDHIT is determined by sequence length. It is actually pretty straightforward:

1. Sort all sequences by length in decreasing order. Longest sequence first, shortest sequence last.
2. Take the first sequence (longest sequence) as the representative of the first cluster.
3. Compare the rest of the sequences with the first sequence.
4. If any of the sequence falls above the threshold for similarity, then it is grouped together with the first sequence.
5. Subsequent comparisons have to fall above the threshold for similarity with all sequences in this cluster group.
6. After this first round of cluster, the next ungrouped sequence is set as the representative of the next cluster.
7. Repeat step 1-5 with this new cluster group.

The output cluster sequences are the longest sequence out of each cluster group.

Short word filter

CDHIT doesn't use dynamic programming to determine sequence similarity. That's probably the biggest reason for it's speed. It looks strictly at exact sequence identity of k-mers.

More specifically, it looks at k-mer size of 2-5. For example here is a 20 base pair overlapping window of two sequences being compared by CDHIT:

```
position:  01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20
sequenceA: A  G  T  C  A  A  A  T  G  G  C  A  T  A  G  G  A  T  A  T
           |  |  x  |  |  |  |  |  |  x  |  |  |  |  |  |  |  |  |  |
sequenceB: A  G  A  C  A  A  A  T  G  A  C  A  T  A  G  G  A  T  A  T
2-mer      1        2  3  4  5  6        7  8  9  10 11 12 13 14 15
3-mer               1  2  3  4           5  6  7  8  9  10 11 12
4-mer               1  2  3              4  5  6  7  8  9  10
5-mer               1  2                 3  4  5  6  7  8
```

The two sequences, A and B, compared over this 20 base pair window has 2 mismatches. Meaning the identity is 90%.

Using some complicated maths, the exact number of matching k-mers needed for a certain identity threshold is determined by:

```
L - K + 1 - (1 - p) * K * L
```

Where L is the window length, which is basically length of the shorter sequence being compared. K is the size of the k-mer. P is the identity threshold. Using this equation, the number of matching 2-mer for an identity of 90% over a 20 base pair window is:

```
20 - 2 + 1 - (1 - 0.9) * 2 * 20 = 15
```

So we need at least 15 matching 2-mers to say the two sequences within this window have 90% identity. The required numbers for other k-mers can also be determined with this equation.

Disadvantages

There are two cases where CDHIT will performed badly. The first has to do with the short word filter technique. If the mismatches between two sequences are distributed evenly, then the number of higher k-mers can potentially be artifically low.

Take this example:

```
position:  01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20
sequenceA: A  G  T  C  A  A  A  T  G  G  C  A  T  A  G  G  A  T  A  T
           |  |  x  |  |  x  |  |  x  |  |  x  |  |  x  |  |  x  |  |
sequenceB: A  G  A  C  A  A  A  T  G  A  C  A  T  A  G  G  A  T  A  T
2-mer      1        2        3        5        7        8        9
3-mer
4-mer
5-mer
```

Because the mismatches are evently distrubuted, there are zero k-mers for k > 2. However, cases of evenly distributed mismatches are extremely rare, as real biological sequences usually have motifs.

The second case where CDHIT can perform wrongly has to do with the the greedy incremental algorithm where the order of clustering is determine by sequence length.

Let's take the example of two sequences, A and B. A and B do not cluster together. A is longer than B, so A is used first for clustering.Subsequent sequences X and Y both pass the similarity threshold when compared to A, so they are grouped with A. However, X and Y actually matches B better; but because the order is determined by length, X and Y get's grouped with A.

Summary

The main advantage of using CDHIT is speed. There are some caveats to using CDHIT in terms of resolution of similarity scoring. However, these concerns are rare in biological sequences due to their non-random nature. There are many flavors of CDHIT meant for various types of data sets described in the user manual.