CROP v1.33 Quick Guide

1. What’s New?

v1.33:

- Fixed bugs sometimes resulting in strange results as mentioned in **Jorge Barriuso, Jose R Valverde and Rafael P Mellado (2011)**

v1.32:

- Introduced a new parameter –r (look at parameter section for more details)

v1.31:

- Fixed bugs in reading fasta files with “\r\n” instead of “\n” as end of line and with “,” in sequence names.

v1.3:

- Optimized performance for diverse data (number of underlying OTUs is huge).
- Fixed a bug in pair-wise alignment program which may result in segmentation fault.

v1.22:

- Changed the way users define their standard deviation to support automated script.
- Fixed a bug which resulted in incorrect fragmentation of .fasta file.
- Dynamically change block size to accelerate.

v1.21:

- Fixed a bug which resulted in program crash when –*b* parameter is set too large.

v1.2:

- Now user can define their own standard deviation interval by using –*t* parameter.
- OpenMP applied thus the program can be run on multiple CPUs.
- Output format changes.

2. How to compile CROP?

CROP needs GSL (GNU Scientific Library). Please follow the link below to install GSL first and then compile CROP. There are three .h files included in bayesianclustering.h, which are gsl_rng.h, gsl_math.h, gsl_randist.h
Link for GSL: http://www.gnu.org/software/gsl/
For gcc compiler, -m64 parameter is optional, add it to Makefile if you want a 64-bit version of CROP

3. How to run CROP v1.32

CROP is a command-line program, which currently supports both Linux and Windows system. In order to run CROP, simply download the executable files and use the following command:

CROP –i MyInput.fasta

4. Parameters

CROP has below parameters which are able to tune by user.

-i: This parameter specifies the name of the input file. Currently CROP only support fasta format file as its input.

-o: This parameter specifies the name of the output file. If not specified, as in the above example, then MyInput.fasta.out will be used.

-b: This parameter specifies the number of blocks. CROP will split the original input file into blocks with same size to improve efficiency. Default value is $\frac{\text{Number of sequences}}{\text{The number specified by -z parameter}}$.

-e: This parameter specifies the number of iterations of MCMC. Default value is 2000. Increase this value to enhance accuracy (recommended value is at least 10*block size).

-g, -s, -l and -u: These parameters specify the similarity level of CROP's clustering. ***–g*** corresponds to 95% , ***-s*** corresponds to 97% and ***–l and -u*** allows user to define their own restriction intervals, where ***–l*** specifies lower bound and ***–u*** specifies upper bound

-m: This parameter specifies the maximum number of "split and merge" process to run. Default value is 20, which is also the maximum allowed.

-z: This parameter specifies the maximum number of sequences to be dealt after the initial "split and merge". Default value is 500, which means each block of sequences contains 500 sequences at most. WE HIGHLY suggest this parameter be adjusted for different data and different computer since it could affect efficiency significantly. For further details about optimizing parameters for specific data, please refer to **Section 6.**

-r: This parameter controls the size of the clusters to be considered as "rare" in last merging using bc step. Basically, in order to enhance the performance of CROP on highly diverse data, we tend to separate the clusters into 2 groups, "abundant" and "rare". "abundant" clusters will be clustered first, then we map all the "rare" clusters to these "abundant" ones, and finally those "rare" cannot be mapped will be clustered separately. So if we set –r 5, then all the clusters with size <=5 will be considered "rare" in above procedure and **–r 0 will yield the best accuracy.** If you believe your data is not too diverse to be handled, then –r 0 will be the best choice. Default value for –r is 2.

A complete command line looks like this:

CROP –i MyInput.fasta –o MyOuput –b 200 –e 3000 –g –m 15 –z 1000 –r 1

5. Output Format

CROP V1.2 generates three output files: .cluster, .cluster.list, .cluster.fasta

Below are several dummy outputs:

.cluster:

```
2                    //the number of OTUs
Alpha 30 2.32
```

Beta 27 2.18 //Alpha, Beta denote the names of the center sequence of each cluster, 30 and 27 are cluster sizes, 2.32 and 2.18 denote the Gaussian standard deviation

.cluster.list:
Alpha *A1,A2,...,A30*
Beta *B1,B2,...B27* //This part outputs the contents of each cluster

.cluster.fasta:
>Alpha
ACGTTACA
>Beta
AGCCTTGG //This part outputs the sequence of each center in fasta format

6. Choice of Parameters

In order to run CROP efficiently, it is important to tune parameters for specific data set.
There are several parameters that are essential for efficiency:

-b:

This parameter, as specified above, defines the number of blocks to be used for the **first round** only. In other words, ***-b*** determines the size of blocks in the first round of clustering.
For very large data set, it is important to make the first round as fast as possible thus effectively reduce the number of sequences needs to be clustered in the following rounds.
Hint of choosing *–b*:
Our experience showed that ***–b*** should be chosen such that each block in the first round contains about 50 sequences. i.e.: for a data set containing 100,000 unique sequences, we set ***–b 2000***

-z:

This parameter, as specified above, defines the size of blocks to be used for **all rounds** (if ***–b*** is specified, then ***–z*** will not affect the **first round**).
For data set with different average sequence length, this parameter should be tuned such that it won't take too long for each block to do pairwise alignment.
Hint of choosing *–z*:
Our experience showed that ***–z*** should be chosen such that if ***–z=Z***, then ***ZL<150,000***, where ***L*** is the average length of the sequences. i.e.:
For a data set of full length 16S rRNA sequences (~1,200 – 1,400 bp), we can set ***–z 100***
For a data set of V6 region sequences only (~60 bp), we can set ***–z 2000***
Notice: Generally, the smaller the –z, the faster the program will run. However if –z is too small, then it's possible that the clusters cannot shuffle and merge well thus extend the number of rounds needed or result in a large block to cluster in the final step (which indeed compromise efficiency). Thus the choosing of –z is kind of tricky. Another way of

experimentally test which –z should be chosen is sample several small subsets of original data with different sizes and run CROP with –b 1 and find out the largest size at which CROP can run in a reasonable time (i.e.: <15min), and specify the value for –z parameter.

-l and –u:

We provide a list of suggested choices for these two parameters:

Choice of parameters	**Corresponding cut-off threshold**
l=0.2 u=0.5	**1%**
l=0.5 u=1.0	**2%**
l=1.0 u=1.5	**3%**
l=1.5 u=2.5	**5%**
l=3 u=4	**8%**
l=4 u=5	**10%**
Not supported, please contact author	>10%