Draft genome assembly of lichenized *Nostoc* from *Peltigera membranacea*

Hákon Jónsson

Raunvísindadeild
Háskóli Íslands
2010

Draft genome assembly of lichenized *Nostoc* from *Peltigera membranacea*

Hákon Jónsson

10 eininga ritgerð sem er hluti af
Baccalaureus Scientiarum gráðu í stærðfræði

Leiðbeinendur
Dr. Ólafur S. Andrésson
Dr. Zophonías O. Jónsson

Raunvísindadeild
Verkfræði- og náttúruvísindasvið
Háskóli Íslands
Reykjavík, Apríl 2010

Draft genome assembly of lichenized *Nostoc* from *Peltigera membranacea*.
10 eininga ritgerð sem er hlutif af *Baccalaureus Scientiarum gráðu* í stærðfræði.

Raunvísindadeild
Verkfræði- og náttúruvísindasvið
Háskóli Íslands
Hjarðarhaga 2-6
107 Reykjavík
Sími: 525 4000

Skráningarupplýsingar:
Hákon Jónsson, 2010, Draft genome assembly of lichenized *Nostoc* from *Peltigera membranacea*,
BS ritgerð, Raunvísindadeild, Háskóli Íslands, 17 bls.

Prentun: Háskólaprent
Reykjavík, Apríl 2010

1 Abstract

Lichens are a combination of a fungus and a photosynthetic partner in a symbosis. In the case of *Peltigera membranacea* the photosynthetic partner is a cyanobacterium.

This thesis is about an approach to assemble the *Nostoc* genome of the lichen species *Peltigera membranacea* based on parallel sequencing (454, Solexa and etc.) of an uncultured sample.

A contig is a continuous sequence of nucleotides bases A,C,G,T assembled from overlapping DNA fragments. If the contigs are sorted in a descending size order and we sum up to the half of the total length of the contigs, the length of the last contig at half of the total length is the N-50 contig size.

We had disappointing results with the initial assembly using all the 454-data with the proprietary assembly program Newbler. The number of contigs was high and the N-50 contig size was low, compared to the amount of sequencing.

The solution we found was to sort the 454-reads and Newbler contigs using Blast[1] with known *Nostoc* genomes, then assemble the sorted reads using Mira[2]. This is a novel method since we are assembling multiple genomes without separating individual species before sequencing.

Contents

2 Introduction to the project

The aim of the project is to produce a draft assembly of the *Nostoc* part of the lichen species *Peltigera membranacea*. The sample that was used for the DNA-sequencing was taken from the field (Keldur). This complicates the assembly since we have to assemble several genomes (*Nostoc*, fungal mitochondrial and nucleus) using a sample which also contains a substantial amount of soil bacteria.

The basic idea behind large scale parallel DNA-sequencing, is that the genome DNA-sequence comes in overlapping DNA-fragments which you need a program to assemble together. The mean size of the DNA-reads from the 454-sequencing is around 350 bases but the reads from the solexa-sequencing are around 36 bases. The solexa-reads are good to verify single nucleotide conflicts but the 454-reads are fine for draft contigs but have problems with homopolymers. Thus we used a mixture of these two sequencing methods.

The initial sequencing was done using 454-sequencing of thallus material (fungus and *Nostoc*). Later a lichen sample with mostly the apothecial part (sporeforming fungus, mostly fungus) was sequenced using 454 and Solexa-sequencing, this was done to obtain a higher proportion of fungal reads. The number of reads from the sequencing is given by the following table (1).

Speciment	454 nr.	Solexa nr.
1	2,262,921	-
2	2,166,858	41,457,296

Table 1: *Peltigera membranacea* reads

As the table shows we have a lot of data. The estimated genome sizes are around 40 M.b. for the fungus and 9 M.b. for the *Nostoc*. For an assembly with all the *P.mem* data we would probably want a computer with around 50 gb of memory, but our main computer only has 8 gb of memory. This complicates things even more. Prokaria, who did the 454 sequencing, did a time consuming assembly of all the 454-reads using Newbler on a computer with 22 gb of memory.

The statistics from the Newbler assembly of all the 454 reads from *P.mem.* are given in table (2). As the table shows the N-50 contig size of the assembly is quite low and the number of contigs is

Stats.	P.mem.
Number of contigs	79,343
Total consensus	119,543,184
Avg. contig size	1,506
N-50 contig size	2,470
Largest contig size	57,449

Table 2: Newbler contigs over 500 bases from 454-reads

high. In light of these results a different method is needed to construct a draft genome for *P. mem*.

The solution that we found was to sort the sequences using the program Blast[1] by homology with known genomes from the *Nostoc* family, then assemble the reads that have homology to the *Nostoc* genomes. This leaves primarily fungal reads. The rest of this thesis is about this sorting and assembling.

3 Sorting the *Nostoc* reads

The following figure (1) and list explain the algorithm of sorting and assembling for the *Nostoc* 454-reads.

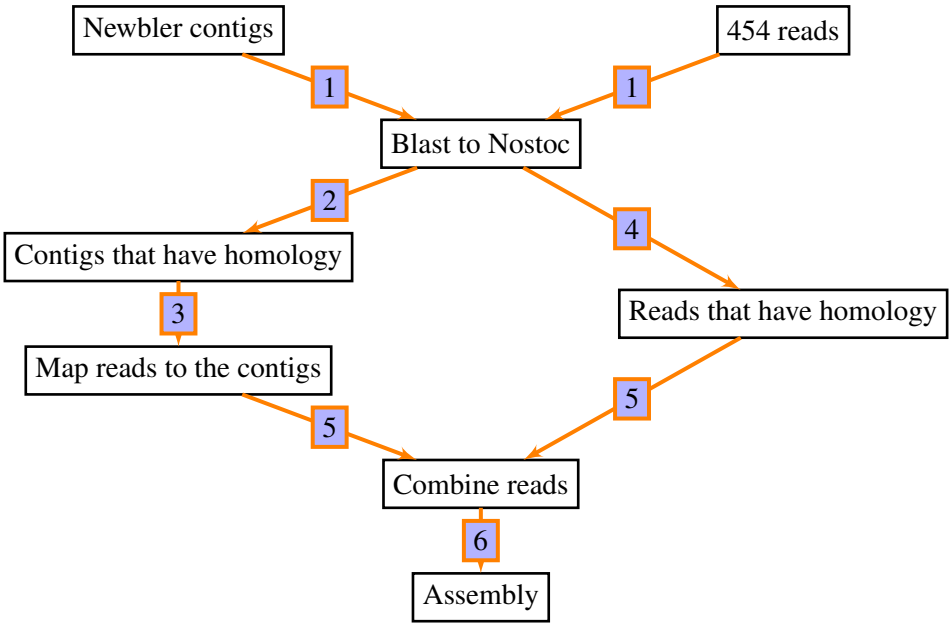

Figure 1: Assembly of *P.mem. Nostoc* genome

The following list corresponds to the blue markers in figure (1). In the following list I call a Blast[1]-hit, the length of the homology between the query and subject sequence in the output from the Blast[1] program with default settings for nucleotides sequences as described in chapter (7.1).

1: The Newbler contigs and 454 reads are blasted against the *Nostoc* genomes.

2: The Newbler contigs that have Blast[1]-hits over a minimum specified length.

3: All of the 454-reads are aligned to the Newbler contigs that have Blast[1]-hits over specific length. This was done using the program Consed[3], we used the default settings for 454-reads reference assembly in Consed[3] version 19.

   Reference assembly is when you match DNA sequences to a reference DNA sequence

4: The 454-reads that have Blast[1]-hits over a minium specific length.

5: The reads from matched *Nostoc* Newbler contigs and the single 454-reads with homology to *Nostoc* are combined. The duplicate reads from this process are of course reduced to one copy.

6: The *Nostoc* 454-reads are then passed to the assembler Mira[2] using various length cutoffs on the Blast[1]-hits in step 2 and 4.

We used the following *Nostoc* genomes (available in Genbank[1]) from the *Nostoc* family for this algorithm.

[1] http://www.ncbi.nlm.nih.gov/Genbank/

Nostoc punctiforme, ATCC 29133

Anabaena sp. (strain *PCC 7120*), ATCC 27893

Anabaena variabilis, ATCC 29413

The idea behind this method is to assemble a new DNA sequence using the Newbler contigs and the sorted *Nostoc* 454 reads for DNA sequences that already exist in the reference *Nostoc* genomes.

3.1 Results of blast

Distribution of the blast hits is given by table (3) from using Newbler contigs over 500 bases as a query and the *Nostoc* genomes as subject. The default settings for nucleotide reads was used for the Blast[1] program, the settings are in chapter (7.1).

	P.mem.
Hits length	Nr. of hits
10 - 50	234,706
50 - 100	72,800
100 - 200	58,069
200 - 300	11,803
300 - 400	2,266
400 - 500	778
500 - 600	507
600 - 700	379
700 - 800	243
800 - 900	209
900 - 1,000	209
1,000 - 2,000	813
2,000 - 3,000	356
3,000 - 4,000	250
4,000 - 5,000	174
5,000 - 6,000	109
6,000 - 7,000	101
7,000 - 8,000	68
8,000 - 9,000	53
9,000 - 10,000	48
10,000 - 20,000	239
20,000 - 30,000	38
30,000 - 40,000	0
40,000 - 50,000	10

Table 3: Blast hits of Newbler *P. mem.* contigs to *Nostoc*

We did the same for the 454-reads the results are in table (4).

In tables (3) and (4), the second column is the number of Blast[1]-hits in the specific length interval. Multiple hits can belong to the same DNA-sequence and the DNA-sequences can have homology to more than one genome. There may be a high degree of redundancy in the three reference sortings, therefore the sorted reads are reduced to unique reads before assembly.

	P.mem.
Hits length	Nr. of hits
10 - 20	66,136,179
20 - 30	9,415,851
30 - 40	1,771,761
40 - 50	422,855
50 - 60	219,014
60 - 70	143,064
70 - 80	101,250
80 - 90	79,314
90 - 100	60,532
100 - 200	339,747
200 - 300	221,964
300 - 400	226,771
400 - 500	139,661
500 - 600	12,035
600 - 700	7

Table 4: Blast hits of *P. mem.* 454-reads to *Nostoc*

4 Assembly of the sorted *Nostoc* reads

We used the program Mira[2] to assemble the 454-reads. The main reasons for this are:

It can use the traceinfo (orginal raw sequence data) of the 454 data, and the assigned quality values for the bases.

It can do a hybrid assembly using Solexa and 454 data. Most assembly programs can only use one type of sequencing data.

The program is highly customizable and it is open source.

We used different blast mininum hit length cutoffs for the assembly, to find the optimal assembly. The optimal assembly would be the one that had the highest genome coverage of *Nostoc puntiforme*, fewest contigs, greatest N50 and median size of contigs.

We ran the Mira[2] with a draft genome option to find the near optimal length cutoffs. Then we ran the most promising assemblies again with the accurate option in Mira[2]. The settings for Mira[2] are in (7.2). The following list explains the stats in table (5).

N-X contig size:
Where X is an integer in the interval $[0, 100]$. If the contigs are sorted in a descending size order, we sum up to X-percentage of the total length of the contigs, length of the last contig until X-percentage of the total length is the N-X contig size.
Query ratio:
The ratio of sequence of *Nostoc puntiforme* that the contigs cover. This was done by blasting the contigs against the *Nostoc* genome. The length of the part of the *Nostoc* genome that has homology to the contigs is divided by the total length of the *Nostoc* genome, this is the Query ratio. The default settings in the Blast[1] programs for nucleotides reads was used, see chapter (7.1) for settings.

		500-contig	700-contig	900-contig	2000-contig	3000-contig
30-read	Number of contigs	2,115	1,964	1,995	2,042	2,079
	Total consensus	10,140,877	9,846,287	9,786,879	9,450,626	9,316,003
	Largest contig	94,555	97,088	87,125	89,427	89,428
	N-50 contig size	17,332	19,022	18,725	17,782	16,694
	N-90 contig size	1,539	1,575	1,559	1,459	1,435
	N-95 contig size	1,010	1,022	1,015	981	981
	Query ratio	0.59	0.58	0.57	0.57	0.58
50-read	Number of contigs	2,076	1,950	1,931	1,934	1,997
	Total consensus	10,044,609	9,774,116	9,634,338	9,243,737	9,064,033
	Largest contig	89,426	85,034	89,429	89,425	89,430
	N-50 contig size	17,840	18,511	18,087	18,443	17,693
	N-90 contig size	1,553	1,584	1,558	1,500	1,423
	N-95 contig size	1,029	1,052	1,027	1,001	972
	Query ratio	0.59	0.57	0.57	0.56	0.57
100-read	Number of contigs	2063	1,930	1,878	1,901	1,928
	Total consensus	9,983,000	9,672,405	9,504,490	9,036,176	8,785,168
	Largest contig	89,425	87,136	89,429	89,421	87,133
	N-50 contig size	17,117	18,164	18,559	16,828	17,245
	N-90 contig size	1,553	1,569	1,580	1,484	1,432
	N-95 contig size	1,019	1,045	1,021	983	977
	Query ratio	0.59	0.56	0.58	0.55	0.55

Table 5: Mira[2] draft contigs that are over 500 bases and have an average coverage over 7.

We ran the assembly with the 900-Contig-100-Read[2] and 700-Contig-30-Read again since those assemblies seemed to have good combination of large contigs and good query ratio of the *Nostoc puntiforme* genome.

	700-Contig-30-Read	900-Contig-100-Read
Number of contigs	1,787	1,680
Total consensus	9,912,559	9,515,585
Largest contig	151,652	163,817
N50 contig size	24,088	24,089
N90 contig size	1,679	1,693
N95 contig size	1,086	1,087
Query ratio	0.56	0.56

Table 6: Mira[2] contigs that are over 500 bases and average coverage over 7 from second run.

These assemblies are quite similar, so choosing the 900-Contig-100-Read for future work wouldn't matter much and probably has higher quality. The total difference of the consensus from 700-Contig-30-Read and 900-Contig-100-Read is around 0.5 mega bases, this suggest if we lower stringency of the filtering we get a longer total consensus.

[2]Newbler and 454 reads length Blast[1]-hit cutoff

	Newbler contigs from all 454-data	Newbler contigs from 900-Contig-100-Read	Mira[2] contigs from 900-Contig-100-Read
Number of contigs	79,343	1,857	6,312
Total consensus	119,543,184	8,199,089	13,490,484
Avg. contig size	1,506	4,415	2,128
N-50 contig size	2,470	13,382	9,225
Largest contig size	57,449	61,067	163,817

Table 7: Mira[2] and Newbler contigs over 500 bases comparsion

To make a rough comparsion with the Newbler contigs from the all of the 454-data in table (2) we extracted contigs over 500 bases from the 900-Contig-100-Read run with Mira[2]. We also did a new assembly with Newbler with the reads from the 900-Contig-100-Read group. The results are in table (7).

The comparsion in table (7) shows increased largest contigs size and N-50 contig size. The mean size of the contigs doesn't say much since there is a large quantity of small contigs that skew the mean size. The Newbler contigs from all the 454-data are from a different version than the Newbler contigs from 900-Contig-100-Read, details about this are in chapter (7.3).

5 Conclusion

Sorting reads is an essential part of assembling genomes from a mixed genome sample. This assembly would be much harder if we didn't have the reference *Nostoc* genomes to sort out the *Nostoc* reads.

As table (5) shows, finding the optimal blast cutoff is dependent on whether you want more reads at the expense of letting non-*Nostoc* reads into the assembly. The point is, the optimal assembly depends on your requirements for the assembly.

Table (7) suggest that the sorting of the 454-reads improves the assembly. Table (5) shows there is room for improvement of the sorting but it's time consuming to find optimal parameters since with our computer each assembly with Mira[2] takes around 6 hours for this amount of data.

6 Description of programs

6.1 Sort_reads_from_contigs_in_ace_files_by_list.pl

```
1  #! /usr/bin/perl
2  # ================================================================
3  # Author:         Hakon Jonsson
4  # Name:           Sort_reads_from_contigs_in_ace_files_by_list.pl
5  # Description:    Picks out the reads that in selected contigs
6  # ================================================================
7
8  use Getopt::Long;
9
10 ##################################################################
11 #                      Parameter parsing
```

```
12 ######################################################################
13
14 Usage() if (@ARGV == 0 or !GetOptions('list=s' => \$list_file,
15                                       'ace=s' => \$ace_file_name,
16                                            'h' => \$Help,
17                                                ));
18 &Usage if $Help;
19
20 ######################################################################
21 #                    Explains how to use the program
22 ######################################################################
23
24 sub Usage
25 {
26   print "Unknown option: @_\n" if ( @_ );
27   print "Usage: $0 -list List_file -ace Ace_file\n";
28   exit;
29 }
30
31 ######################################################################
32 #                    Running through the list
33 ######################################################################
34
35 open LIST, $list_file;
36 while (<LIST>){   #Loads the list file into a hash
37   chomp;
38   s/^>//;
39   $List{$_}=1;
40 }
41
42 ######################################################################
43 #                    Running through the Ace file
44 ######################################################################
45
46 open ACEFILE, $ace_file_name;
47 {
48   my( %comp,
49       @contig_line, @read_line, @Compl_line,
50       $crap, $turn_on_reading,
51       $read_name_for_comp_info, $comp_info, $Bases);
52   while (<ACEFILE>){
53     chomp;
54     if (/^CO /){
55       @contig_line=split;
56       ($crap, $contig_name)=@contig_line;
57       if ($List{$contig_name}){
58         #If the contig name is in the list file
```

```
59            $turn_on_reading++;
60          } else {
61            $turn_on_reading=0;
62          }
63        }
64        if (/^AF/ && $turn_on_reading) {
65          #Checks the AF line to see if the sequence is completed
66          @Compl_line=split;
67          ($crap,$read_name_for_comp_info,$comp_info)=@Compl_line;
68          $comp{$read_name_for_comp_info}=$comp_info;
69        }
70        if (/^RD / && $turn_on_reading){
71          $Bases="";
72          $revBases="";
73          @read_line=split;
74          ($crap,$read_name)=@read_line;
75          $_=<ACEFILE>;
76          #Get rid of the sequence name line
77          while(/^[\w\*]/){
78            #Gets the bases
79            chomp;
80            s/\*//g;
81            $Bases.=$_;
82            $_=<ACEFILE>;
83          }
84          if (!$comp{$read_name}) {
85            print ">".$read_name."\n".$Bases."\n";
86          } else{
87            $revBases=&Complement($Bases);
88            print ">".$read_name."\n".$revBases."\n";
89          }
90        }
91      }
92    }
93
94    ######################################################################
95    #                        Complements DNA sequence
96    ######################################################################
97
98    sub Complement {
99      my ($COMPDNA);
100     my($DNA)=@_;
101     @DNA = split( '', $DNA );
102     foreach my $nuc (reverse @DNA){
103       #Reverse the @DNA array
104       if ($nuc =~ /a/) {
105         $COMPDNA.='t';
```

```
106       }elsif ($nuc =~ /A/) {
107           $COMPDNA.='T';
108       }elsif ($nuc =~ /t/) {
109           $COMPDNA.='a';
110       }elsif ($nuc =~ /T/) {
111           $COMPDNA.='A';
112       }elsif ($nuc =~ /g/) {
113           $COMPDNA.='c';
114       }elsif ($nuc =~ /G/) {
115           $COMPDNA.='C';
116       }elsif ($nuc =~ /c/) {
117           $COMPDNA.='g';
118       }elsif ($nuc =~ /C/) {
119           $COMPDNA.='G';
120       }else {
121            die "$0:␣␣Bad␣nucleotide␣in␣subroutine␣complement␣␣[$nuc]\n
                ";
122       }
123     }
124     return $COMPDNA;
125 }
```

Description:

The program loads a file with contig names, then goes through an ACE file. This ACE file describes how the contigs are put together from overlapping DNA fragments. If we find a contig in ACE file that has a matching name in the list then we send the reads that make up the contig to the STDOUT.

The program is written in Perl.

7 Settings and info for programs

7.1 Blast[1]

The default settings for the blastall program in the Blast[1] are shown here. For the nucleotide reads the program was called with -p blastn.

```
blastall 2.2.19   arguments:

  -p  Program Name [String]
  -d  Database [String]
    default = nr
  -i  Query File [File In]
    default = stdin
  -e  Expectation value (E) [Real]
    default = 10.0
  -m  alignment view options:
0 = pairwise,
1 = query-anchored showing identities,
```

```
2 = query-anchored no identities,
3 = flat query-anchored, show identities,
4 = flat query-anchored, no identities,
5 = query-anchored no identities and blunt ends,
6 = flat query-anchored, no identities and blunt ends,
7 = XML Blast output,
8 = tabular,
9 tabular with comment lines
10 ASN, text
11 ASN, binary [Integer]
    default = 0
    range from 0 to 11
  -o  BLAST report Output File [File Out]  Optional
    default = stdout
  -F  Filter query sequence (DUST with blastn, SEG with others) [String]
    default = T
  -G  Cost to open a gap (-1 invokes default behavior) [Integer]
    default = -1
  -E  Cost to extend a gap (-1 invokes default behavior) [Integer]
    default = -1
  -X  X dropoff value for gapped alignment (in bits) (zero invokes default behavior)
      blastn 30, megablast 20, tblastx 0, all others 15 [Integer]
    default = 0
  -I  Show GI's in deflines [T/F]
    default = F
  -q  Penalty for a nucleotide mismatch (blastn only) [Integer]
    default = -3
  -r  Reward for a nucleotide match (blastn only) [Integer]
    default = 1
  -v  Number of database sequences to show one-line descriptions for (V) [Integer]
    default = 500
  -b  Number of database sequence to show alignments for (B) [Integer]
    default = 250
  -f  Threshold for extending hits, default if zero
      blastp 11, blastn 0, blastx 12, tblastn 13
      tblastx 13, megablast 0 [Real]
    default = 0
  -g  Perform gapped alignment (not available with tblastx) [T/F]
    default = T
  -Q  Query Genetic code to use [Integer]
    default = 1
  -D  DB Genetic code (for tblast[nx] only) [Integer]
    default = 1
  -a  Number of processors to use [Integer]
    default = 1
  -O  SeqAlign file [File Out]  Optional
  -J  Believe the query defline [T/F]
```

```
  default = F
-M  Matrix [String]
  default = BLOSUM62
-W  Word size, default if zero (blastn 11, megablast 28, all others 3) [Integer]
  default = 0
-z  Effective length of the database (use zero for the real size) [Real]
  default = 0
-K  Number of best hits from a region to keep. Off by default.
    If used a value of 100 is recommended.
    Very high values of -v or -b is also suggested [Integer]
  default = 0
-P  0 for multiple hit, 1 for single hit (does not apply to blastn) [Integer]
  default = 0
-Y  Effective length of the search space (use zero for the real size) [Real]
  default = 0
-S  Query strands to search against database (for blast[nx], and tblastx)
     3 is both, 1 is top, 2 is bottom [Integer]
  default = 3
-T  Produce HTML output [T/F]
  default = F
-l  Restrict search of database to list of GI's [String]  Optional
-U  Use lower case filtering of FASTA sequence [T/F]  Optional
-y  X dropoff value for ungapped extensions in bits (0.0 invokes default behavior)
    blastn 20, megablast 10, all others 7 [Real]
  default = 0.0
-Z  X dropoff value for final gapped alignment in bits (0.0 invokes default behavior)
    blastn/megablast 100, tblastx 0, all others 25 [Integer]
  default = 0
-R  PSI-TBLASTN checkpoint file [File In]  Optional
-n  MegaBlast search [T/F]
  default = F
-L  Location on query sequence [String]  Optional
-A  Multiple Hits window size,
    default if zero (blastn/megablast 0, all others 40 [Integer]
  default = 0
-w  Frame shift penalty (OOF algorithm for blastx) [Integer]
  default = 0
-t  Length of the largest intron allowed in a translated nucleotide
    sequence when linking multiple distinct alignments.
    (0 invokes default behavior; a negative value disables linking.) [Integer]
  default = 0
-B  Number of concatenated queries, for blastn and tblastn [Integer]  Optional
  default = 0
-V  Force use of the legacy BLAST engine [T/F]  Optional
  default = F
-C  Use composition-based score adjustments for blastp or tblastn:
    As first character:
```

```
    D or d: default (equivalent to T)
    0 or F or f: no composition-based statistics
    2 or T or t: Composition-based score adjustments as in Bioinformatics 21:902-911,
    1: Composition-based statistics as in NAR 29:2994-3005, 2001
        2005, conditioned on sequence properties
    3: Composition-based score adjustment as in Bioinformatics 21:902-911,
        2005, unconditionally
    For programs other than tblastn, must either be absent or be D, F or 0.
         As second character, if first character is equivalent to 1, 2, or 3:
    U or u: unified p-value combining alignment
    p-value and compositional p-value in round 1 only [String]
  default = D
-s  Compute locally optimal Smith-Waterman alignments (This option is only
    available for gapped tblastn.) [T/F]
  default = F
```

7.2 Mira[2]

We used Mira[2] version 3 release candidate 4 for the assemblies. We had two different settings, that is one for the draft contigs:

```
mira_3rc4_dev_linux-gnu_x86_64_static/bin/mira
--project=assembly --job=denovo,genome,draft,454 -AS:urd=no -SK:not=2
```

For the accurate contigs we used:

```
mira_3rc4_dev_linux-gnu_x86_64_static/bin/mira
--project=assembly --job=denovo,genome,accurate,454 -AS:urd=no -SK:not=2
```

The -AS:urd=no option is to turn off the uniform read coverage error correction since the sorting of the *Nostoc* reads can skew the coverage.

7.3 Newbler

We used the default settings for the assembly of all the 454 reads, version 2.0.01.14. Four months later when we were finished with the sorting of the *Nostoc* reads a new version 2.3 of Newbler was released, we used that version for the *Nostoc* Newbler assembly.

References

[1] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers and David J. Lipman, (1990), Basic Local Alignment Search Tool. J. Mol. Biol. in 1990 pp 403-410

[2] Chevreux, B., Wetter, T. and Suhai, S. (1999), Genome Sequence Assembly Using Trace Signals and Additional Sequence Information. Computer Science and Biology: Proceedings of the German Conference on Bioinformatics (GCB) 99, pp. 45-56.

[3] David Gordon, Chris Abajian and Phil Green (1998), Consed: A Graphical Tool for Sequence Finishing. Genome Research p 195-202.