OMBlast Software Manual

An alignment tool for optical mapping data

(Version 1.0, January 2016)

Alden Leung

This document is the software manual that accompanies the program OMBlast. Details about its algorithm can be found in the published manuscript and not this document.

The software is available from https://github.com/aldenleung/OMBlast and is released under a GPL license (see the software distribution for details).

Minimum Requirements

Linux or Windows operating system (Software has been tested on Ubuntu 14.10 and Microsoft Windows 7.).
Java 1.7 or above.

Installation

All required libraries are placed in the folder `lib/`.

Quick steps:

1. Compile the OMBlast package in the OMBlast folder:
   ```
   javac -d bin -sourcepath src -cp "lib/*" @classes
   ```
2. Build a runnable jar file for OMBlast:
   ```
   jar cvfm OMBlast.jar manifest -C bin .
   ```
3. Run OMBlast:
   ```
   java -jar OMBlast.jar
   ```
4. Run other scripts:
   (Linux) `java -cp "bin:lib/*" aldenjava.script.ScriptName`
   (Windows) `java -cp "bin;lib/*" aldenjava.script.ScriptName`

Quick Commands

Basic Command

```
java -jar OMBlast.jar --refmapin reference.ref --optmapin
data.data --optresout result.oma
```

Additional options:

- **Setting up multi-thread**
  `--thread 1`
- **Alter different modes in processing alignment**
  `--filtermode 1 --postjoinmode 2 --clustermode 1`
- **Change the output filtering parameters**
  `--minclusterscore 30 --minconf 0.4 --maxclusteritem 1`

Sample Run

java -jar OMBlast.jar --refmapin ./Reference/Ecoli.ref
--optmapin ./WithoutSVData/Ecoli_mid_1.sdata --optresout Ecoli_mid_1_result.omd

Optical Maps Filtering

Optical maps which are short or contain few signals are usually aligned in low confidence, as they carry less information. Users could take an optional step to filter the optical maps by their size and their number of signals. We recommend users remove an optical map with 100 kbp in size and 10 signals.

File formats

Since there is no standard file format for optical mapping data, OMBlast supports 9 optical map data formats.

The file formats of input and output files will be automatically determined from the file extension, if the argument of input format is set to -1 or if there is no file format specified.

1. Optical mapping data format (input)

Most file formats were developed by various research groups for optical maps generated from either OpGen or Bionano. Each file contains tuples to represent a single or multiple optical maps. However, details of their respective file specifications are lacking. OMBlast supports the following file formats:

- ref, fa01, spots, data, sdata, bnx, cmap, opt, silico, xml

2. Optical mapping alignment format (output)

Each alignment method has its own output format. Here, we only provide a list of optical map alignment format we currently support as input:

- oma, omd, xmap, val, soma, psl

and as output:

- oma, omd, xmap

The oma and omd formats were created for OMBlast. Interested users can refer to the oma/omd file specification in a separate file [OpticalMapFileSpecification.docx]. RefAligner generates xmap files for the alignment results. Since Valouev and SOMA do not specify a file extension for their output, we name their output as val and soma, respectively. TWIN generates psl as their final output.

Command line options

1. Input

Reference input options

refmapin	Input reference file
refmapinformat	Input reference file format. See file format section for details.

Query input options

optmapin	Input query file
optmapinformat	Input query file format. See file format section for details.

Result input options

optresin	Input result file
optresinformat	Input result file format. See file format section for details

OMBlast requires a reference optical map file and a query optical map file as inputs. The result file is optional for realignment purpose. Please refer to the File Formats section for details.

2. Module options

Query filtering module

minsig	Minimum number of signals for the query
minsize	Minimum size for the query
exactmatch	Allow exact match of query ID to the reference ID

This is one step of pre-alignment filtering of the query, according to minimum number of signals and size. Optionally, users should consider setting a more stringent value for *minsig* and *minsize*, if they find the alignment results are not specific enough. This is especially useful if the size of the reference optical map is small. The option *exactmatch* should only be set to false when applying pairwise alignment with the same optical map file as input and output. This option ensures that the query will not exactly match itself.

Scoring parameters

m	Score for matching signal
fpp	Penalty score for false positive signal (extra cut)
fnp	Penalty score for false negative signal (missing cut)

Scoring parameters are a crucial factor to the final results. The default score for *m*, *fpp*, and *fnp* are determined from our previous analysis (5, 2 and 2 respectively). This combination of scores is robust enough for data with a wide range of error rates. With very good data quality, users could consider increase *fpp* and *fnp*.

Penalty parameters for joining partial alignments

indelp	Penalty score for joining partial alignments with indel relationships
Invp	Penalty score for joining partial alignments with inversion relationships
transp	Penalty score for joining partial alignments with transposition relationship

The penalty score for joining partial alignments is arbitrary. The basic idea is to join two partial alignments only if both of them are confident enough. From our experience, it is difficult to strike a balance between specificity and sensitivity. Setting the penalty score higher increases the specificity of non-SV-containing queries but lowers the sensitivity of SV-containing queries. While a default penalty score is provided based on our empirical analyses, if users want to fine tune their results for sensitivity and specificity, we recommend users start by spotting several regions on the reference with confident SV that can be retrieved using a lower penalty score, and then extract the SV-containing queries aligned within these regions. Users can then increase the penalty score and observe whether the SV-containing queries could still be properly aligned.

Error tolerance parameters

meas	Maximum measurement error tolerated
ear	Maximum scaling error tolerated
falselimit	Maximum consecutive false positive / negative signals
maxseedno	Maximum similar seeds on the query

Error tolerance parameters are strictly related to the final results. The default value for *meas* and *ear* are determined from our previously analysis. Optionally, users can tune the tolerance parameters according to the instrument used for optical maps generation. However, the effects from altering *meas* and *ear* are not straight-forward. While OMBlast could miss the true alignment if *meas* and *ear* are set too small, OMBlast could also locate the wrong alignment if *meas* and *ear* are set too large. For advanced users, we recommend deduce the two parameters by learning the nature of optical maps specific to the instrument by using the alignment results from a standard data set (e.g. *E. coli*).

Seed-and-extend

seedingmode	Score for matching signal
k	Kmer / K-tuple length
maxnosignal	Maximum length for “no signal” region for seeding
local	Allow local alignment in an extension

The *seedingmode* is set to -1 for automatic mode selection according to the *k*, which is default to 3 and yields results with both high sensitivity and specificity in general. One could consider modifying *k* if the data quality is very high, or they are aligning very long optical mapping contigs to reference. The parameter *local* is set to true by default to indicate that local alignments are allowed. Users could change this value to false to force OMBlast to perform global alignment. This also makes OMBlast penalize missing and extra signals outside the aligned regions.

Redundant alignment removal

postjoinmode	0: Disable postjoin mode 1: Remove alignment of same start and stop position and Cigar string 2: Remove redundant alignment by dynamic programming

To remove the redundant partial alignment, one could simply remove partial alignments which are exactly the same as selecting 1 for *postjoinmode*. Alternatively, redundancy can be removed through dynamic programming by selecting 2 for *postjoinmode*.

Alignment filtering

filtermode	0: Disable filtering 1: Filter with all alignment filtering options below 2: Filter with minscore only
minmatch	Minimum number of matches
maxfp	Maximum number of false positive signals
maxfn	Maximum number of false negative signals
maxfpr	Maximum false positive rate
maxfnr	Maximum false negative rate
minscore	Minimum score
minsubfragratio	Minimum ratio of aligned segments
minsigratio	Minimum ratio of aligned signals
trimmode	0: Disable trimming during filtering 1: Enable trimming during filtering
maxtrim	Maximum trimming steps done in filtering

Alignment filtering is a module that is applied to the partial alignments. The basic idea of filtering is to remove partial alignments with low quality. The quality of partial alignment could be attributed to the number of matched signals, the number and ratio of extra and missing signals and the aligned portion of the query. This default quality requirement is lenient. We recommend that users raise these requirements if exceptionally high non-specific alignments are observed.

In certain situations, an alignment could have a local region with low quality. We may only want to remove the regions of low quality rather than discard the whole partial alignment. By setting *trimmode* to 1 to enable the trimming sub-module, partial alignments are trimmed on the left or right with a maximum steps as stated in *maxtrim*. This ensures that the trimmed resulting partial alignments could meet the quality requirement.

We recommend users to set a more stringent *minscore* if they align a large query to a large reference, for example, contigs alignment; or if they want to enable *clustermode* 3 in alignment joining.

Alignment joining

clustermode	0: Disable partial alignment joining 1: Partial alignment joining with indel relationships 2: Partial alignment joining with indel and inversion relationships 3: Partial alignment joining with indel, inversion and translocation relationship
closeref	Maximum distance between reference locations of two alignments classified as nearby to the reference location
closefrag	Maximum distance between reference locations of two alignments classified as nearby to the query location
trimear	Keep track of the scaling error tolerance during trimming
maxtrim	Maximum trimming steps for joining two partial alignments
clusterlocalpenalty	Calculate penalty for local alignment

In general, we recommend users use *clustermode* 1 to attain both specificity and sensitivity. If users are willing to handle inversion, they can also select *clustermode* 2 but specificity would lower slightly for queries without inversion. The use of *clustermode* 3 takes more time as partial alignments from any location could be joined. We only recommend running *clustermode* 3 on high-throughput optical maps for realignment only, where users provide an alignment file describing a list of filtered but potentially aligned regions of the queries, to shorten the running time.

In case of a large query versus a large reference, for example, in silico digested scaffolds on optical mapping assembly, we recommend users use *clustermode* 3. The value of *maxtrim* should be set much higher to 50 as the overlapping regions of partial alignments are much larger in practice.

Joined alignment filtering

minclusterscore	Minimum score
minconf	Minimum confidence
mincluserfragratio	Minimum aligned segment ratio
minclustersigratio	Minimum aligned signal ratio
maxclusteritem	Maximum number of outputs
overlapcluster	Allow overlapping alignments of multiple outputs

Users could change the value here for final results filtering. For the definition of score and confidence, please see the published manuscript for more information. Here we only describe the practical settings of these two values. The default value of *minclusterscore* could already be applied to alignments of optical maps from a wide range of species. However, from our experience, users should consider tuning this value according to the signal density of the reference, where *minclusterscore* could be set lower for a reference with low signal density. Confidence is an arbitrary cut-off for removing non-specific alignments. If users want to capture duplicated or highly repeated regions, or output multiple alignments, *minconf* should be set lower, or even to 0. We note that short optical maps are usually aligned with low score and confidence. Thus, one needs to decrease the *minclusterscore* and *minconf* to capture short optical maps. However, we would like to remind users that with fewer signals, short optical maps contain less information and hence their alignments are inaccurate. The users need to trade-off precision against recall. The *maxclusteritem* parameter controls the number of final alignments to be output, while *overlapcluster* is set to false to prohibit multiple alignments from overlapping. Users

Multi-thread option

thread	Maximum number of threads for alignment

3. Output

Result output module

optresout	Filename of output result file
optresoutformat	Output result file format. See file format section for details
writeunmap	Output “unmapped” alignments if no possible alignment is available for a query (applicable to the OMA and OMD file formats)
multiple	Output multiple alignments or output the first alignment
writeinfo	Output query information in the result file (OMA and OMD file formats only)

Please refer to the File Format section for details on the result file format. By default *multiple* is set to true, and we do not recommend users to change the value. To control the number of entries that are output, users can refer to the parameters in the alignment joining module.

The value *writeinfo* is set to true by default. However, if users want to output multiple alignment entries, saving each entry with query information could lead to a very large file size. By setting *writeinfo* as false, the file size will shrink but users need to supplement the result file with the optical map data file for analysis.