Automated Oligonucleotide Design Pipeline
`aodp` ver. ”2.5.0.1”
user manual

Manuel Zahariev (mz@alumni.sfu.ca)
Wen Chen (wen.chen@agr.gc.ca)

June 9, 2018

Terms

Acknowledgement

This work is financially supported by Agriculture & Agri-Food Canada (AAFC) funded project #97 and Canadian Safety and Security Program (`CSSP`) funded project CRTI 09-462RD/CSSP 30vv01.

Background

The Automated Oligonucleotide Design Pipeline (`aodp`) is an utility that generates oligonucleotide signatures for sequences in the *FASTA* format and for groups in a phylogeny in the *Newick tree* format. `aodp` version 2 (and higher versions) was developed by Manuel Zahariev, based on the following previous work:

- `sigoli`: an utility that provides the ability to efficiently identify oligonucleotides for sequences and groups of sequences from large databases. *Sigoli* was developed by Manuel Zahariev with support from Agriculture and Agri-Food Canada, Simon Fraser University and a number of individuals and other organizations between 2001-2005.
- `aodp-1.3`: a wrapper around the *sigoli* utility, providing consolidation and of the output from multiple runs of *sigoli* and validation using *blast* against sequence databases. `aodp-1.3` was developed by Kaisheng (Kevin) Chen, Chuyang (Charles) Qi, Eric Hardy and Wen Chen with support from Agriculture and Agri-Food Canada.

Contents

1 Components

In this document aodp refers to aodp version 2 (or higher), unless explicitly specified.
aodp contains the following components:

- aodp core: command-line application written in C++
- clado: simple perl wrapper around Bio::Tree::Draw::Cladogram (Bioperl module).
- clus: utility providing taxon presence information from a list of experimental results of matches against oligo cluster signature probes.

Both utilities can be invoked from the command line.
aodp-xref can also be invoked from aodp by using the --xref command-line switch.

2 Prerequisites

Synopsis (*Ubuntu*):

```
$ sudo apt-get install g++ make perl bioperl blast2
```

The ./configure script will generate errors for missing prerequisites.

2.1 System requirements

- aodp should run on most *UNIX* systems. It has been tested on *Ubuntu 14.04*.
- On systems with multiple cores aodp will automatically run in multithreaded mode, maximizing resource usage.
- System memory is the main limitation for the maximum size of the input data set. Data sets (*FASTA* format) of up to 10MB have been processed successfully on a machine with 8GB of RAM. Larger data sets result in disk/memory swapping, which increases dramatically the running time.

2.2 C++ 11

- aodp is built from source using *C++ 11*
- The C++ standard library and associated utilities (e.g. make) are also needed.

On *Ubuntu*:

```
$ sudo apt-get install g++ make
```

2.3 perl/BioPerl

BioPerl is a collection of Perl modules that facilitate the development of Perl scripts for bioinformatics applications.
clado and the --cladogram option of aodp use the following components:

- perl version 5.8 or better
- bioperl

On *Ubuntu*:

```
$ sudo apt-get install perl bioperl
```

3 Installation

Synopsis:

```
$ tar -xzf aodp-x.tar.gz
$ cd aodp-x
$ ./configure
$ make
$ sudo make install
$ man aodp
```

The latest version of aodp is available at the following locations:

- https://bitbucket.org/aafc-mbb/aodp_release
- https://github.com/AAFC-MBB/AODP_releases
- http://lifeintel.org

Download the latest aodp distribution (e.g. aodp-x.tar.gz).
You may verify the integrity of the distrubution by comparing its digital signature with the one published on the site in the file aodp-x.sha256 corresponding to your downloaded version. Example:

```
$ sha256sum -b aodp-2.0.1.tar.gz
09a8307c5079f1beb93a0d13bea5aa5d397dedb2b6625950d2a2f0659962fcff
*aodp-2.0.1.tar.gz
```

```
$ cat aodp-2.0.1.sha256
09a8307c5079f1beb93a0d13bea5aa5d397dedb2b6625950d2a2f0659962fcff
*aodp-2.0.1.tar.gz
```

Unpack the distribution archive, then enter the resulting directory. Example:

```
$ tar -xzf aodp-x.tar.gz
$ cd aodp-x
```

Run the configuration script:

```
$ ./configure
```

Build `aodp`:

```
$ make
```

As a user with `root` privileges, install `aodp`. Example:

```
$ sudo make install
```

If everything went well, you will be able to see the `aodp` manual:

```
$ man aodp
```

As a user with `root` privileges, you may uninstall `aodp` when you no longer need it:

```
$ sudo make uninstall
```

3.1 Installation in other directories

By default, the `aodp` install script will try to:

- Install the `aodp` executables (`aodp` and `aodp-xref`) in `/usr/local/bin`.
- Install the `aodp` man pages in `/usr/share/man/man1`.

If you would like to point the installation to other directories (for example if you do not have root access to the machine), pass the following parameters to `./configure`:

- `BIN_TARGET_DIR=/your/location/for/executables`.
- `MAN_TARGET_DIR=/your/location/for/manpages`.

Example:

```
./configure BIN_TARGET_DIR=/home/myuser/bin \
MAN_TARGET_DIR=/home/myuser/man/man1
```

Unless the directories you have specified are already in your path (and respectively `MANPATH`), you will need to specify the full path when accessing them.
For example, call `aodp`:

```
/home/myuser/bin/aodp --strings Anisogramma.fasta
```

Open the `aodp` manpage:

```
man /home/myuser/man/man1/aodp.1.gz
```

4 aodp

4.1 SYNOPSIS

aodp [*options*] *output fasta-sequence-file...*

4.2 DESCRIPTION

aodp generates oligonucleotide signatures for sequences in **FASTA** format and for all groups in a phylogeny in the **Newick tree** format.

4.3 INPUT

Multiple input sequence files in the **FASTA** format can be specified anywhere on the command line. The usual file name wildcards can be used. Any command line argument without the prefix -- will be treated as a file name.
Each **FASTA** file can contain multiple sequences. Sequence identifiers are read from the **FASTA** description lines (lines starting with >). Everything on the description line following a space is ignored. Multiple occurrences of sequences with the same identifier are considered as the same sequence, with several discountiguous sections.
If any of the specified *fasta-sequence-file*-s cannot be open or completely parsed, **aodp** will terminate with an error.

4.4 OUTPUT

At least one output option is required. For each output option, if **file-name** is not specified, the result is written to standard output.

--strings[=file-name]

Writes all oligo strings grouped by sequence or group

--positions[=file-name]

Output for Array Designer (tab-separated list of oligo sites)

--ranges[=file-name]

All ranges of sites of oligos grouped by sequence or group

--fasta[=file-name]

All oligos in FASTA format

--gff[=file-name]

All oligos in GFF format

--newick[=file-name]

Writes a phylogeny (**Newick tree** format), with exactly the same structure as the input tree, with generated labels for internal nodes.

Requires a **--tree-file** input

--node-list[=file-name]

List of sequences for every internal node in the phylogeny

Requires a **--tree-file** input

--lineage[=file-name]

Lineage of every sequence in the phylogeny

Requires a **--tree-file** input

--fold[=file-name]

Predicted secondary structure and calculated two-state melting temperature for all oligonucleotide signature candidates discarded because of higher melting temperature than **--max-melting**

If the option **--max-melting** is not specified, will print the predicted secondary structure for all oligonucleotide candidates. Melting temperatures below 0C or over 100C will be indicated as `*`.

--cladogram=(file-name)

printout in the eps format of a cladogram associated with the annotated phylogeny tree. All nodes with identified oligos are marked with a `*` and these nodes and their descendents are coloured red.

Requires a **--tree-file** option and a file-name

--time[=file-name]

Tab-separated `user`, `system`, `elapsed` time and maximum memory usage for various phases of processing

--basename=(name)

All oligos in the following formats:

- `name.oligo.strings` strings grouped by sequence and/or group
- `name.oligo.fasta` **FASTA**
- `name.oligo.gff` **GFF**
- `name.oligo.tab` tab-separated
- `name.oligo.positions` positions
- `name.oligo.ranges` ranges

If the option **--tree-file** is specified (phylogeny tree), the following output files will also be generated:

- `name.newick` phylogeny tree with labeled internal nodes
- `name.node-list` list of sequences for every internal node
- `name.lineage` lineage of every sequence in the phylogeny
- `name.cladogram.eps` cladogram associated with the annotated phylogeny tree

The output options **--strings**, **--fasta**, **--gff**, **--tab**, **--positions**, **--ranges --newick**, **--node-list**, **--lineage** and **--cladogram** are incompatible with **--basename**.

--cluster-list[=file-name]

Generates the list of clusters: groupings of sequences that can be identified by at least one oligonucleotide signature (cluster oligonucleotide signature).

The output contains the following tab-separated columns:

- Numeric identifier of the cluster. This is a generated value.
- Space-separated list of all identifiers of sequences contained in the cluster
- If the cluster matches exactly a target node (leaf node or internal phylogeny node), an additional column with the name of this node is included

--cluster-oligos[=file-name]

Generates the list of all oligonucleotide signatures for all clusters identified. The output contains the following tab-separated columns:

- Numeric identifier of the cluster. This matches the value in **--cluster-list**.
- String representation of the cluster oligonucleotide signature

--clusters=(name)

Generates a file of cluster nodes (`name.cluster-list`) and a file of cluster oligonucleotide signatures (`name.cluster-oligos`)

The output options **--cluster-list** and **--cluster-oligos** are incompatible with **--clusters**.

--match=(target-FASTA-file)

Finds the closest matching source sequence for each sequence from the **target-FASTA-file**. Works as follows:

- Build the smallest set of sequences (`minimum-set`) that can explain the largest portion of each target sequence based on matching cluster oligonucleotide signatures.
- Align (using a modified Needleman-Wunsch global alignment algorithm) each sequence in the `minimum-set` with the target sequence and calculate the percentage similarity. Multiple source sequences may have the same percentage similarity with the target sequence.

For each source sequence with maximum percentage similarity to the target sequence, prints to standard output or to the file specified by **--match-output** the following tab-delimited fields:

- Target sequence identifier
- Source sequence identifier or **-** if no matching source sequence was found
- Percentage similarity
- Length (bp) of the portion of the target sequence that matches perfectly the source sequence
- Length of the target sequence
- Size of the `minimum-set` (affects the running time)
- Size of the `maximum-set` of sequences contained in any clusters matched by the target sequence

Suggestion: **--max-homolo=0** must be specified, otherwise matches containing homopolymers will be ignored and the match percentage will be lowered

--match-output=(output-file)

Redirect the output from **--match** to the output file

Requires a **--match** input

4.5 OPTIONS

Other command line parameters are optional.

--help

Display this command summary then exit.

--version

Display version and copyright information

--oligo-size=(size[-size])

Look for oligonucleotide signatures of sizes within the specified range

--tree-file=(relative-file-name)

Use the phylogeny file in the **Newick tree** format that groups the sequences in the input; oligos will also be sought for all nodes in the phylogeny

--outgroup-file=(relative-file-name)

The outgroup list is a case-sensitive list of species (sequence names) that are to be excluded from the final output. They will still be used in the generation of oligos, but oligos specific to them will be omitted.

Will terminate with an error if a sequence name in the **--outgroup-file** is not found in any *fasta-sequence-file*.

--isolation-file=(relative-file-name)

A list of taxa or sequences to isolate (one item per line). Sequences whose name match (case-sensitive) as a complete substring any of the items in the **--isolation-file** will be the targets of the oligo search. Only oligos for sequences that match items on the list or nodes that have sequences that match items on the list will be sought. Individual entries in the **--isolation-file** may match more than one sequence. For example, `carotovorum` will match the following sequences:

- `Pectobacterium_carotovorum_actinidae`
- `Pectobacterium_carotovorum_brasiliense`

Will terminate with an error if an item in the **--isolation-file** does not match any sequence in any *fasta-sequence-file*.

--database=(file)

Validate the resulting oligos against a reference database in the **FASTA** format. This option requires specifying a corresponding **--taxonomy** option.

Will terminate with an error if the database file contains sequence names that are not specified in the taxonomy file.

Requires a **--taxonomy** input.

--taxonomy=(file)

Taxonomy information associated with the sequences in the reference database. Each line has tab-separated sequence name and lineage, ending in species name (`s__Genus_species_...`).

The species name (Genus species) of all input sequences encoded as `XX_9999_Genus_species_...` will be extracted. Oligos for input sequences that don't match any sequence or match only sequences in the reference database for the same species will be kept (super-specific oligos); sequence oligos that match reference sequences with no species name or with another species name will be discarded.

Oligos for internal nodes (group oligos) that match reference sequences with a species name other than any of the species names in the group (from the input sequences) will also be discarded.

Requires a **--database** input.

--ignore-SNP

Single polymorphism sites (SNP) will be ignored in the design of oligos. **--ignore-SNP** will generate less oligos in more time

--reverse-complement

Will take into account the reverse complement of all sequences (default=no):

- SO will also be generated for the reverse complement of each sequence
- All generated SO will be validated against all direct and reverse complement of all sequences

--crowded=(yes/no)

Indicates whether for the **--ranges** and **--positions** outputs an oligo range is populated with intermediary sites (default no)

```
         |<--range with signature oligos-->|

                  middle of range
    =====|================|================|=====  nucleotide sequence

         *                *                *       --crowded=no
         * *    *    *    *    *    *    * *       --crowded=yes>
          |                      |
    --first-site-gap    --inter-site-gap
```

--first-site-gap=(gap-size)

For a **--crowded** display, the size of the gap between the border of the range and the first interior site (default 5)

--inter-site-gap=(gap-size)

For a **--crowded** display, indicates the size of the gap between sites inside an oligo range (default 5).

This parameter cannot be set to 0.

--ambiguous-sources=(yes/no)

Whether sequences containing ambiguities are considered in the analysis. The names of these sequences will be written to a file called excluded.fasta in the current directory; default yes

--threads=(count)

The maximum number of threads for multiprocessor systems. By default, aodp will detect the number of available processors/cores (n) and will use n-1 threads, or one thread on single processor systems.

--max-ambiguities=(count)

Indicates the maximum number of ambiguous bases (default 5). Sequences with more than this number of ambiguous bases will not be included in further processing. Their names will be written to a file called **excluded.fasta** in the current directory. If this parameter is not specified, no sequences will be filtered out based on the number of their ambiguous bases.

--max-crowded-ambiguities=(count)

Indicates the maximum number of ambiguous bases within an oligo size. Sequences with more than this number of ambiguous bases anywhere within a window will not be included in further pro cessing. Their names will be written to a file called excluded.fasta in the current directory. If this parameter is not specified, no sequences will be filtered out based on the number of their ambiguous bases.

--ambiguous-oligos=(yes/no)

Whether oligos containing ambiguous bases will be sought in the analysis; default `no`

--max-homolo=(size)

Maximum length of a homopolymer (e.g. only `As`) in any oligo; default `4`; `0` means no oligos with homopolymers will be removed

--max-melting=(temperature-C)

Maximum melting temperature (Celsius) for any discovered oligo. Oligos with higher melting temperature will be removed from the result. If this option is not specified, all oligos are reported.

The two-state melting temperature is calculated using the NN model (SantaLucia and Hicks 2004), applied to the most stable single-strand self-folding configuration at temperature **--max-melting**.

```
  ----------------------------------------------
 | Tm = DH x ( DS + R x ln ( CT / x )) + 273.15 |
  ----------------------------------------------
       (SantaLucia and Hicks 2004; eq. 3)
```

Use **--salt** to specify the salt (NaCl) concentration and --strand to specify the strand concentration (CT). In melting temperature calculations for oligonucleotide signatures, x is always `1`.

Since a given strand may have a number of competing self-folding configurations, the actual melting temperature will be consistently lower for multi-state coupled equlibria than the calculated two-state melting temperature.

The options **--max-melting** and **--ambiguous-oligos=yes** are incompatible.

--salt=(Na+ concentration in M)

Na+ concentration (default `1M`). Valid values are between `0.05M` and `1.1M`

The **Na+** concentration is used in entropy calculations:

```
  -----------------------------------------------
 | DS[Na+] = DS[1M NaCl] x 0.368 x N/2 x ln[Na+] |
  -----------------------------------------------
       (SantaLucia and Hicks 2004; eq. 5)
```

`N` is the total number of phosphates in the folded configuration. For self-folding configurations, `N` is the strand length in nucleotides minus `1`.

--strand=(single strand concentration in `mM`)

strand concentration (default `0.1`) in mM used in (SantaLucia and Hicks 2004; eq. 3). Valid values are between `0.01` and `100`.

In melting temperature calculations for oligonucleotide signatures, x is ALWAYS `1`.

4.6 EXAMPLES

By default, output from aodp is directed to the standard console:

```
          output type
           =======
     $ aodp --strings Anisogramma.fasta
                      =================
                       input database
```

Wildcards are supported for the files in the input database:

```
     $ aodp --strings fasta/*
```

By specifying a phylogeny tree in the Newick format, oligonucleotide signatures will also be sought for internal nodes of the phylogeny:

```
     $ aodp --strings --tree-file=Anisogramma.tre Anisogramma.fasta
                                  ===============
                       phylogeny tree for input database
```

A number of output types (in separate files with the same prefix) will be automatically generated by using the option --basename:

```
                       output file prefix
                       ==================
     $ aodp --basename=Anisogramma.output Anisogramma.fasta
```

The target oligonucleotide signature length (in base pairs) can be specified as a number:

```
     $ aodp --strings --oligo-size=32 Anisogramma.fasta
```

...or as a range:

```
     $ aodp --strings --oligo-size=24-32 Anisogramma.fasta
```

Oligonucleotide signatures containing ambiguous base pairs can be sought:

```
     $ aodp --strings --ambiguous-oligos=yes Anisogramma.fasta
```

Sequences with ambiguities in the input database can be filtered out using the options **--ambiguous-sources**, **--max-ambiguities** and **--max-crowded-ambiguities**:

```
     $ aodp --strings --max-crowded-ambiguities=8 Anisogramma.fasta
```

Specific sequences or groups of sequences can be excluded from the output using an *outgroup* file:

```
     $ aodp --strings --outgroup-file=A.outgroup Anisogramma.fasta
```

The output can be restricted to only sequences specified within an *isolation* file:

```
$ aodp --strings --isolation-file=A.iso Anisogramma.fasta
```

The output can be filtered for maximum melting temperature (Celsius):

```
$ aodp --strings --oligo-size=48 --max-melting=45 Anisogramma.fasta
```

Additional parametes (strand concentration in mM and salt concentration in M) for the calculation of the melting temperature can be included:

```
$ aodp --strings --oligo-size=48 --max-melting=45 Anisogramma.fasta
       --salt=0.1 --strand=2
```

Actual melting temperature and folding configuration for the resulting oligonucleotide signatures can be displayed:

```
$ aodp --strings --oligo-size=48 --max-melting=45 Anisogramma.fasta
       --fold --salt=0.1 --strand=2
```

The output of aodp can be cross-referenced against a reference database, by using the **--taxonomy** and **--database** options:

```
$ aodp --tree-file=Anisogramma.tre Anisogramma.fasta
       --database=UNITE_public_mothur_full_10.09.2014
       --taxonomy=UNITE_public_mothur_full_10.09.2014_taxonomy.txt
```

4.7 SEE ALSO

clado

generates an eps file from a phylogeny tree file in a Newick format. Nodes that have names ending in ⋆ are colored red.

clus

interprets the results of matches of experimental samples against cluster oligo signatures generated by aodp

DNA thermodynamics

SantaLucia J Jr, Hicks D. 2004. The Thermodynamics of DNA Structural Motifs. Annu. Rev. Biophys. 33:415-40

5 clado

5.1 SYNOPSIS

clado *newick-tree-file output-file*

5.2 DESCRIPTION

clado generates an eps file from a phylogeny tree file in a Newick format. Nodes that have names ending in * are colored red.
clado is a simple wrapper around Bio::Tree::Draw::Cladogram (Bioperl module).

5.3 SEE ALSO

aodp

The Automated Oligonucleotide Design Pipeline finds oligo signatures for sequences and groups of sequences in a sequence database

Bioperl

a collection of Perl modules that facilitate the development of Perl scripts for bioinformatics applications

Bio::Tree::Draw::Cladogram

drawing phylogenetic trees in Encapsulated PostScript (EPS) format (Bioperl module)

Newick tree format

http://evolution.genetics.washington.edu/phylip/newick_doc.html (last retrieved April 14, 2016)

6 clus

6.1 SYNOPSIS

```
clus --cluster-list=Penicillium.cluster-list \
     --cluster-oligos=Penicillium.cluster-oligos \
     --node-list=Penicillium.node-list --matches=sample.matches
```

6.2 DESCRIPTION

clus interprets the results of matches of experimental samples against cluster oligo signatures generated by aodp.
clus is a ”field companion” utility for aodp. It uses as input cluster oligonucleotide signatures generated by aodp and matches them against the result of a test of some of these cluster oligonucleotide signatures against a field sample.

6.3 OPTIONS

--cluster-list=(file)

File containing a list of clusters generated by aodp with the option `--cluster-list`

--cluster-oligo=(file)

File containing a list of cluster oligonucleotide signatures generated by aodp with the option `--cluster-oligos`

--matches=(file)

File containing a list of matches from a field sample against oligonucleotide signatures from the file in `--cluster-oligos`. clus will only consider that items not included in this file were not tested against the field sample. Each row must contain the following tab-separated fields:

- cluster oligonucleotide signature (string representation)
- **1** if this oligonucleotide signature matches the sample and **0** otherwise

--node-list=(file) (optional parameter)

File containing a list of phylogeny nodes generated by aodp with the option `--node-list`

--crop (optional parameter)

Will mark as **not found** all sequences in clusters with oligonucleotide signatures that do not match with the sample: the size of minimum group conclusively present in the sample containing each such sequence is zero.

For example, if a cluster *C* containing sequence *Y* does not match the sample, sequence *Y* will

The smallest group containing sequences will also narrowed by performing set operations on the results.

For example, if the sample matches a cluster oligonucleotide signature for the sequences *X* and *Y*, but the sequence *Y* shows **not found**, `--crop` will show the minimum set for sequence *X* as $\{X\}$.

Without `--crop`, the minimum set for sequence *X* will be shown as $\{X, Y\}$.

6.4 OUTPUT

`clus` prints to standard output sequence-level answers, in rows with the following tab-separated fields:

- sequence identifier (as specified in the second column of the `--cluster-list` file)
- the size of the smallest group conclusively present in the sample that contains the sequence. **0** means that the sequence is conclusively NOT IN the sample; **1** means that the sequence is conclusively in the sample; **more than 1** means that the sequence may or may not be in the sample.
- a list of space-separated sequence identifiers from the smallest group conclusively present in the sample that contains the sequence
- if the group in the previous field is a node in the phylogenetic tree (from `--node-list`), the name of that node is included

6.5 SEE ALSO

aodp

The Automated Oligonucleotide Design Pipeline finds oligonucleotide signatures for sequences and groups of sequences in a sequence database

7 Benchmarks

The running time of `aodp` is influenced by the following factors:

1. Size of the source database
   - For databases without ambiguities, running time should increase linearly with the size of the database
2. Target oligonucleotide size
   - For databases without ambiguities, running time should increase linearly with the maximum target oligonucleotide size
3. Amount of physical memory on the machine
   - If `aodp` uses up all available physical memory, running time increases signifficantly
   - `aodp` has been tested with sequence databases in the *FASTA* format of up to 10MB on a system with 7.2GB
4. Number of ambiguous bases in sequences in the database
   - Performance will be degraded dramatically for databases that have sequences with large numbers of ambiguous bases *within subsequences of the target oligonucleotide size*[1].
   - Use the command-line switch `--max-crowded-ambiguities` to filter such sequences out of the analysis.
   - Alternately, use the command-line switch `--max-ambiguities` to filter out sequences with more than a given number of ambiguous bases.
5. The following command-line options also increase running time:
   - `--diff=no`
   - `--ambiguous-oligos=yes`

Validation of the output using *BLAST+* against a database by using the `--xref` option or the `aodp-xref` utility can increase running time signifficantly.
`aodp` running time benchmarks on a generic system (laptop) have been run on two databases:

- *Bol*: a database of 1,218 aligned genes of the class *Oomycete*, of approximately 2.5MB
- *Fusarium* a database of 10,125 aligned genes from the genus *Fusarium*

The results for a variety of oligonucleotide target sizes in extreme running time scenarios (lower-bound and upper-bound) are reported in Figures 1, 2, 3 and 4. Anomalies in the charts are likely due to variations in CPU speed on the laptop (e.g. power management features).

[1]For example, a subsequence `NNNNNNNNNN` may be the result of a transcription error in the database.

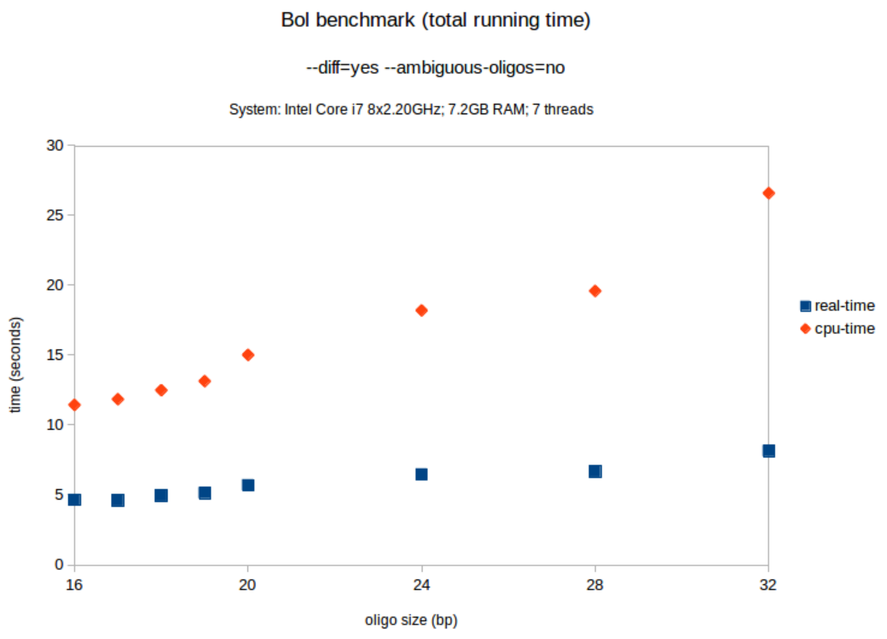

Figure 1: Running time lower bound for database *Bol*

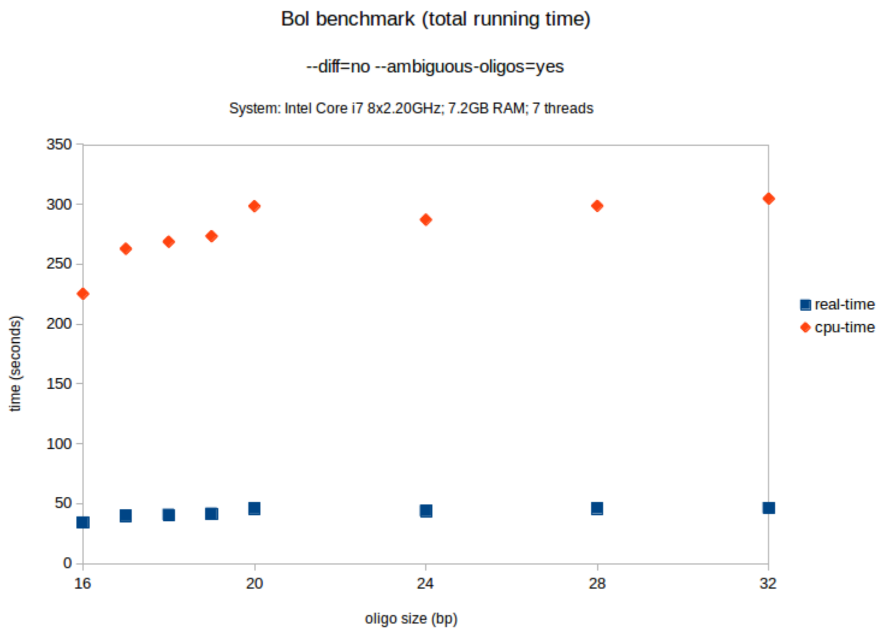

Figure 2: Running time upper bound for database *Bol*

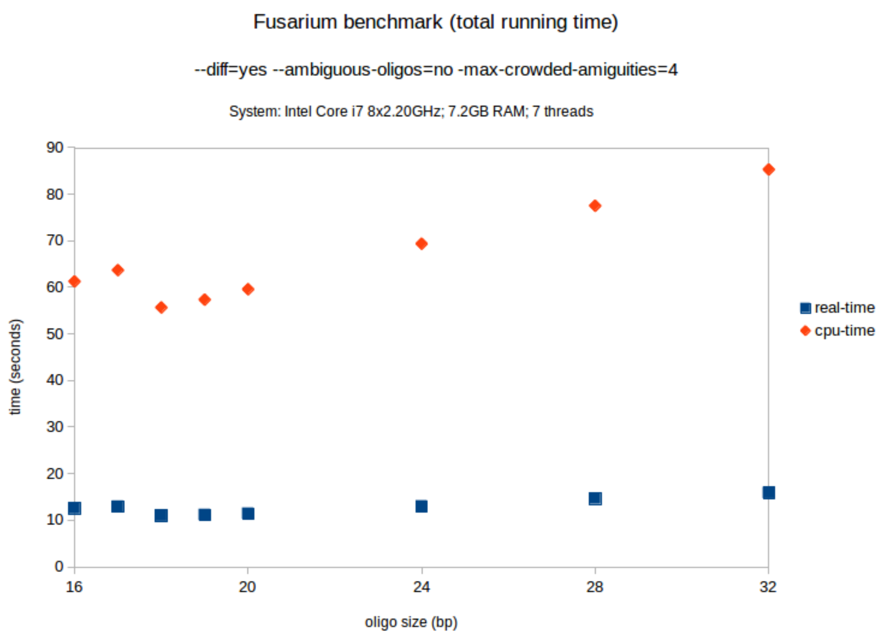

Figure 3: Running time lower bound for database *Fusarium*

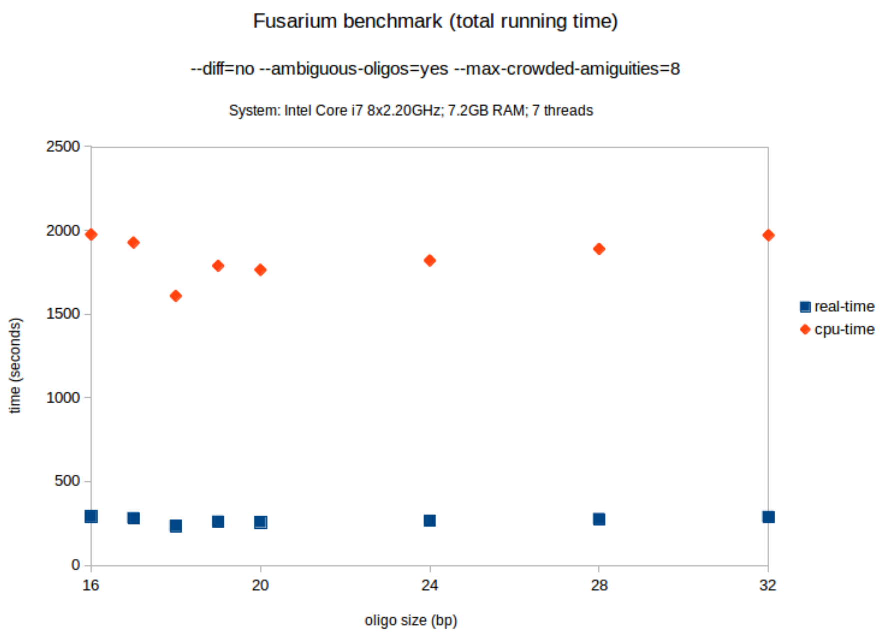

Figure 4: Running time upper bound for database *Fusarium*