Manual i-ADHoRe 3.0

Department of Plant Systems Biology
VIB - Ghent University
Contact: Yves Van de Peer[1]

Internet Based Communication Networks and Servives (IBCN)
Department of Information Technology (INTEC)
Ghent University
Contact: Jan Fostier[2]

March 6, 2012

[1]yves.vandepeer@psb.vib-ugent.be
[2]jan.fostier@intec.ugent.be

Contents

1 Package contents **3**

2 Installation and requirements **4**
2.1 Basic Installation on Linux 4
2.2 Installing CMake 4

3 Using i-ADHoRe **5**
3.1 Usage 5
3.2 Configuration file 5
3.2.1 genome lists 5
3.2.2 BLAST-table 6
3.2.3 Running mode 6
3.2.4 Algorithm parameter settings - general 7
3.2.5 Algorithm parameter settings - Colinear Mode 8
3.2.6 Alignment algorithm 8
3.2.7 Algorithm parameter settings - Cloud Mode 9
3.2.8 Output path 10
3.3 Parallellization 10
3.4 Visualization 10
3.5 Output 11
3.5.1 The multiplicons table 12
3.5.2 The baseclusters table. 13
3.5.3 The anchorpoints table. 13
3.5.4 The segments table. 14
3.5.5 The genes table. 14
3.5.6 The list_elements table. 15
3.5.7 The clouds table 15
3.5.8 The cloud anchorpoint table 16

4 Post processing: i-Visualize **16**
4.1 Example of AlignmentMultiplion.svg 16
4.2 Examples of GHM.png 16

References

Jan Fostier*, Sebastian Proost*, Bart Dhoedt, Yvan Saeys, Piet Demeester, Yves Van de Peer and Klaas Vandepoele (2011) (***** contributed equally)
A Greedy, Graph-Based Algorithm for the Alignment of Multiple Homologous Gene Lists. *Bioinformatics* **27**, 749-756.

Sebastian Proost*, Jan Fostier*, Dieter De Witte, Bart Dhoedt, Piet Demeester, Yves Van de Peer and Klaas Vandepoele (2012) (***** contributed equally)
i-ADHoRe 3.0 Accurate and Sensitive Detection of Genomic Homology in Extremely Large Datasets. *Nucleic Acids Res* **40(2)**, e11.

Previous work:

Simillion, C., Janssens, K., Sterck, L., Van de Peer, Y. (2008)
i-ADHoRe 2.0: An improved tool to detect degenerated genomic homology using genomic profiles. *Bioinformatics* **24**, 127-128.

Simillion, C., Vandepoele, K., Saeys, Y., Van de Peer, Y. (2004)
Building Genomic Profiles for Uncovering Segmental Homology in the Twilight Zone. *Genome Res.* **14**, 1095-1106.

Vandepoele, K., Saeys, Y., Simillion, C., Raes, J., Van de Peer, Y. (2002)
The Automatic Detection of Homologous Regions (ADHoRe) and its application to microcolinearity between Arabidopsis and Rice. *Genome Res.* **12**, 1792-801.

1 Package contents

The i-ADHoRe package consists of the following files and directories:

- `manual.pdf` The documentation file (this file)
- `Makefile` The makefile needed to compile the source code
- `src/` The location of the main C++ source code files
- `src/alignment/` The location of the source code files for the alignment of profiles (consists of 2 methods).
- `src/datastructures/` The location of the source code files for the necessary datastructures.
- `src/debug/` The location of the source code files that were mainly used for debugging purposes.
- `test_set` The location where the files are that are necessary to do the test-run.
- `DatasetI.ini` Dataset I consists of the *Arabidopsis thaliana* genome. This testset should have an approximate running time of 1 minute.
- `DataSetII.ini` Dataset II consists of the *Arabidopsis thaliana*, *Vitis vinifera* and *Populus trichocarpa* genomes. This testset may run for several hours.

2 Installation and requirements

2.1 Basic Installation on Linux

This package requires CMake to build the software. If you don't have CMake already installed, refer to the next section "Installing CMake".
The installation of i-ADHoRe is simple. First, unzip the i-adhore-xxx.tar.gz file (where xxx denotes the version number):

```
tar -xzvf i-adhore-xxx.tar.gz
cd i-adhore-xxx
```

From this directory, run the following commands:

```
mkdir build
cd build
cmake ..
```

A useful option to specify for the cmake command is CMAKE_INSTALL_PREFIX so that you can tell cmake where to install the software.
For example, to install in your local $(HOME)/i-adhore directory you would run:

```
cmake .. -DCMAKE_INSTALL_PREFIX=$(HOME)/i-adhore
```

Afterwards run:

```
make
```

And to install run (as root if necessary):

```
make install
```

It is required that you have a Pthreads library installed. Support for MPI and Googletest unit testing framework is optional.

2.2 Installing CMake

As root, execute the following commands:

- on Redhat / Fedora distributions:

  ```
  yum install cmake
  ```

- on Ubuntu / Debian distributions:

  ```
  aptitude install cmake
  ```

3 Using i-ADHoRe

3.1 Usage

The command-line reads:

```
> i-adhore CONFIGURATIONFILE
```

Where `CONFIGURATIONFILE` is a file containing a list of all files in the dataset as well as all input options and parameters. After reading the settings file and checking if all the necessary options are specified, i-ADHoRe starts detecting colinearity in the dataset. See the DatasetI.ini or DataSetII.ini file, which are included in the i-ADHoRe package, for an example. The configuration file is explained in 3.2.

3.2 Configuration file

3.2.1 genome lists

These are all the genomes that are to be analysed. You can have different fragments (e.g. chromosomes, BACs, contigs,...) per genome, here further referred to as gene lists. They are represented as individual gene list files containing unique gene identifiers, one per line. These identifiers cannot contain spaces and must end by a '`+`' or '`-`' sign, indicating the strand orientation of each gene. Thus, a typical gene list file looks like:

```
gene1+
gene2-
gene3+
gene4+
gene5-
...
```

These list files are listed per genome in the settings file in the following way:

```
genome=some_genome
C01 some_path/first_chrom.lst
C02 some_path/second_chrom.lst
C03 some_path/third_chrom.lst

genome=some_other_genome
C01 another_path/this_chrom.lst
C02 another_path/that_chrom.lst
...
```

You can include as many genomes and lists as you want. Every genome must start with the keyword '`genome=`', followed by an unique genome identifier (no spaces). Next, all the fragments of the genome are listed (one per line) as a unique identifier (unique per genome) followed by a space and the path to the corresponding list file. The list of genome fragments is terminated with a blank line.

3.2.2 BLAST-table

This must be a file containing all gene pairs of the dataset that i-ADHoRe should consider to be homologs. Typically, these are the query and hit pairs from an all-against-all BLAST search on which some selective criteria were applied to filter out non-significant hits, e.g. an e-value cutoff or a more advanced filtering method. These gene pairs should be listed with the same identifiers as used in the gene list files and are separated with a TAB. Note that homology is a symmetrical relationship, thus if the table contains the pair A, B, i-ADHoRe also creates the pair B, A. This can be important if the query-hit pairs are selected by putting an e-value cutoff, since the e-value is not a symmetrical score (BLASTing sequence A against B gives a different result as B against A). A BLAST-table file looks thus like this:

```
gene1  gene2
gene3  gene5
gene6  gene4
...
```

The path to the BLAST-table file is indicated in the settings file as:

```
blast_table=some_path/blast_table.txt
```

For genes clustered into families a different type of table can be used. This setting requires much less memory. So if many genomes are compared in a single run this might be necessary to keep the memory usage within the boundaries of the available hardware. Each gene needs to be in a family even if it's a singleton. To use this option build your BLAST-table like this:

```
gene1  family1
gene2  family1
gene3  family1
gene4  family2
...
```

To indicate you're using this type of table add the following parameter to the settings file.

```
table_type=family
```

3.2.3 Running mode

The algorithm can run in three different modes:

Colinear Mode This is the default mode. It searches for regions of colinearity in the dot matrices between genelists. To add the request for this mode explicitely add the following line to the ini file:

```
cluster_type=colinear
```

Cloud Mode This method looks for regions where gene content is similar but gene order is not necessarily conserved in the two genelists. In the dot matrix this phenomenon will appear as cloud like structures. The distance function used in Cloud mode is the Chebyshev distance. Note that the Cloud mode doesn't perform profile searches . To run the algorithm in Cloud mode add the following line to the ini file:

```
cluster_type=cloud
```

Hybrid Mode To investigate datasets containing a mix of recent and diverged collinear regions the user can choose for the Hybrid mode. In the dot matrices the algorithm first runs in Colinear mode. Then the dots contained in the collinear regions are removed after which the Cloud mode is started. Note that no profile searches are performed in collinear mode! To run the algorithm in Hybrid mode add the following line to the ini file:

```
cluster_type=hybrid
```

3.2.4 Algorithm parameter settings - general

The optimal parameter settings are dependent on the properties of the dataset. An overview is provided in Proost et al. 2011.

Tandem gap size Indicates the maximum Euclidean distance (or diagonal pseudo distance) that can exist between gene pairs to be considered tandem duplicates. This setting is optional, the program will use the default value '`gap_size=`' / 2 if this parameter isn't set. Indicate this value using the identifier '`tandem_gap=`', e.g.

```
tandem_gap=10
```

The probability cutoff A real value between 0 and 1, indicating the maximum probability to be generated by chance a cluster or synthenic cloud can have. Suggested value: 0.001 Indicate this value using the identifier '`prob_cutoff=`', e.g.

```
prob_cutoff=0.001 to 0.01
```

Write Statistics When this parameter is set to true supplemantary statistics are provided which report the percentages of chromosomes that are inside duplicated or collinear regions.

```
write_stats=true/false
```

Note that this doesn't work correct in combination with `flush_output`. (see further) In case this variable is set to true the `flush_output` value should exceed the maximum number of multiplicons in the system.

Limit to level 2 dectection Optionally, the user can force i-ADHoRe to detect only level 2 multiplicons (i.e. multiplicons containing only 2 segments) and not to build profiles. In this case, the program will only perform the basic ADHoRe homology detection. This option is disactived by default. To activate this option, put this line in the settings file:

```
level_2_only=true
```

Multiple hypothesis correction If the same experiment is repeated a number of times the probability to observe rare events increases. To take this into account when the validity of collinear regions or synthenic clouds is evaluated a Multiple Hypothesis Correction is used. The user can choose to not use this, use the Bonferroni method or the False Discovery Rate (FDR) method. This can be done by adding one of the following lines to the ini file (if not added no correction is provided). Suggested value: FDR.

```
multiple_hypothesis_correction=bonferroni, FDR or none
```

3.2.5 Algorithm parameter settings - Colinear Mode

The gap size Indicates the maximum (pseudo-)distance that should exist between points in a cluster. Suggested value: 10-40. Indicate this value using the identifier '`gap_size=`', e.g.

```
gap_size=15
```

Cluster gap size Indicates the maximum (pseudo-)distance that should exist between individual base clusters in a cluster. When i-ADHoRe detects homologous segments, it does so by first detecting base clusters of anchor points in a gene homology matrix. Next, it is checked if several base clusters are located close enough to each other to be grouped into one metacluster. This value should be bigger or equal than the gap size. If unsure, set this to be the same as the gap size. Indicate this value using the identifier '`cluster_gap=`', e.g.

```
cluster_gap=20
```

The Q-value A real value between 0 and 1, indicating the minimum r^2-value (a measure for the linearity of a series of points) a cluster should have. Suggested value: 0.70-0.90 Indicate this value using the identifier '`q_value=`', e.g.

```
q_value=0.9
```

Minimum number of anchor points A whole number indicating the minimum number of anchor points i.e. the number of genes each segment in a multiplicon should have that are homologous to the other segments in that multiplicon. Suggested value: 3-6. Indicate this value using the identifier '`anchor_points=`', e.g.

```
anchor_points=5
```

3.2.6 Alignment algorithm

The user can also choose from 3 alignment algorithms. The progressive Needleman-Wunsch algorithm and the Greedy Graph-based Alignment algorithm. (described in J. Fostier et al. 2011) The latter is called `gg2`, `gg` is the original aligner from i-ADHoRe 2.0. The nw aligner is chosen by default. Suggested setting: gg2.

```
alignment_method=nw, gg, gg2
```

Maximum number of gaps in alignment This is the number of gaps allowed in a segment after aligning it in a multiplicon. This value is always larger than or equal to the `cluster_gap`. Suggested setting: 15-20. Note that this parameter is not influencing the profile search operation.

```
max_gaps_in_alignment=20
```

3.2.7 Algorithm parameter settings - Cloud Mode

Cloud gap size This is in principle the maximum chebyshev distance between two dots in a cloud. The chebyshev distances considers all dots on the edge of a square to be equally distant from the center of mass of the square. In `bruteForceSynthenyMode` the actual chebyshev distance is calculated between dots. In the default case dots are added from the frame around the clouds bounding box, which speeds up the algorithm. The thickness of this frame is in this case the cloud gap. Suggested setting: 10-20

```
cloud_gap_size=20
```

Cloud cluster gap size This is the same as the `cluster_gap`. In case of cloud merging the algorithm guarantees that the two closest points between two clouds are actually closer than this distance removed from each other. Suggested setting: Cloud gap size +5.

```
cloud_cluster_gap=25
```

Cloud filter method To evaluate whether a cloud is generated by chance a statistical method is used. In Cloud mode there can be chosen between 3 filters. One filters according to the number of dots divided by the longest side of the bounding box (density filter). The more relevant filters use the binomial distribution and the binomial distribution corrected for tandem duplicates. (only one dot per row and per column assumed)

```
cloud_filter_method=density, binomial, binomal_corr
```

If no filtermethod is specified in the ini file, the binomial distribution is chosen by default.

bruteForceSynthenyMode When values for the gapsizes are to big (depends on density of the dots in the GHM) there was an avalanche effect spotted. This means that large synthenic clouds are created which keep on merging until eventually only one vaste cloud remains. This effect is due to the fact that the gapsize is used to add new dots within a frame whereby the thickness is equal to the gapsize. This heuristic makes the algorithm faster but it also allows for dots to be added which are in fact not within the gapsize. Gapsizes can be tuned to overcome this effect. Another parameter is included in the input files to study this effect. With this bruteforce method always the actual distance to from a point to a cloud is used and therefore all dots are within the gapsize of another dot.

```
bruteForceSynthenyMode=on,off
```

If not specified in the ini file, the mode is off by default.

3.2.8 Output path

This is the directory to which the output files are written. If the directory does not exist, one will be created. The contents of the different output files will be described in full detail in the next section. The output path is declared like:

```
output_path=mydata/i-adhore_out
```

3.3 Parallellization

There is one parameter related to multithreading:

```
number_of_threads=4
```

The algorithm also supports MPI parallelization. It automatically detects whether the system has MPI support. The number of processes can be specified from the command line:

```
> mpirun -np nProcs i-adhore CONFIGURATIONFILE
```

Hereby nProcs must be substituded with the number of processes, this can be any positive integer number.
The parallelization divides the searches in an intelligent way among the processes and threads. The alignment itself has not been parallelized.

3.4 Visualization

One of the new features of i-ADHoRe 3.0 is that it doesn't require post processing to visualize some of its structures. Two visualization methods have been added. One to visualize the dot matrices and one to visualize the aligned multiplicons.

Visualize Gene Homology Matrix This creates a set of files (for every pair of genelists!) of the form `COLGHM\_genelistx_genelisty` or `SYNTHGHM\_genelistx_genelisty` depending on the fact that it is a matrix from Colinear or Cloud mode. Depending on whether the user has png support (library: libpng) the files will have a .png extension, otherwise they will be written in bmp format.
Dots are drawn in white. If they are part of a significant BaseCluster or Synthenic Cloud they are drawn in yellow. The bounding box of the cluster or cloud is also drawn. For significant clusters it is drawn in green, for the others in red. In Colinear mode the confidence intervals of the BaseClusters is also added by means of blue dots. To turn on dot matrix visualization add the following line to the ini file:

```
visualizeGHM=true/false
```

This variable is set to false by default. An example of a GHM plot is found in the chapter about postprocessing.

VisGPairs This parameter allows for specific genelist pairs to be visualized (dot matrix). Note that `visualizeGHM=false` otherwise this parameter will have no effect.

```
visGPairs=listname1a genome1a listname1b genome1b, listname2a ...
```

Visualize Alignment This creates a set of AlignmentMultipliconID.svg files (for all multiplicons!) containing the visualization of the aligned multiplicons. Hereby `ID` will be substituted with the multiplicon's identifier. When the multiplicon has been rejected this produces a file with only black boxes. Since visualizing all multiplicons might be impossible for large datasets a postprocessor tool is added to the i-ADHoRe package to visualize one multiplicon per turn. The file is a set of rows of coloured boxes. Every row represents a BaseCluster or Segment which has been aligned into the Profile. There are 3 distinct posibilities concerning the boxes.

- White boxes: represent gaps.
- Black boxes: represent genes which have no homologous links with other genes in the profile.
- Coloured boxes: reprensent genes which have a homologous link in the profile (or tandems which are seperated more than the tandem gap size!)

Further on there are lines which link nonaligned genes. If a gene is not aligned (boxes below have different color) and there is no line connecting it to its homologue than this is a gene has a homologue in the same row (segment) and is thus a tandem. To turn on alignment visualization add the following line to the ini file:

```
visualizeAlignment=true/false
```

This variable is set to false by default.

3.5 Output

Verbose If this parameter is set to true the input parameters are printed to the screen, also the ones implicitely set by the program.

```
verbose_output=false
```

Flush output The value of this parameter (default value is 1000) defines when the vector of evaluated multiplicons is written to the outputstream. At the moment 1000 multiplicons are in the vector their properties are written to the output tables described later on. As mentioned earlier, this doesn't work toghether with `write_stats`!

```
flush_output=1000
```

Output tables i-ADHoRe outputs different tables with data as TAB delimited text files to the output directory specified in the settings file. These tables describe the different elements of the multiplicons and clouds that were detected. All of these elements are identified by an unique code to facilitate store of the data in a relational database management system like MySQL or Oracle. In each file, the first line describes the field names.
Below is a description of all the fields of all tables. Note that, for the sake of clarity, the order of the fields described here may be different from the order of the columns in the files.

3.5.1 The multiplicons table

Describes all multiplicons for every multiplication level that was detected. This table is written in the output directory specified in the settings file as `multiplicons.txt`, with the first row indicating the field names. The meaning of the different fields is explained below. Note that this table also lists all level 2 multiplicons that are redundant, meaning that the segments of these multiplicons are included in higher-level multiplicons. These multiplicons are marked as redundant using the **is_redundant** field (see below).

id An unique identifier for each multiplicon.

genome_x, list_x For level 2 multiplicons (i.e. containing only 2 segments), these fields describe the first segment (denoted as the x-segment) of a multiplicon. This segment is identified by the genome it is part of (**genome_x**) and the specific gene list (**list_x**) it is located on.

For multiplicons of level 3 and more, these fields are left blank since for these multiplicons, the collinearity is detected between a profile of the parent multiplicon (indicated in the parent field) and the y segment. Thus for multiplicons with a level of more than 2, the parent multiplicon replaces the x-segment.

parent For multiplicons of level 3 and more, indicates the multiplicon that was used as a profile to detect this multiplicon. The value refers to the **id** field of that multiplicon, which is always a multiplicon of level $n - 1$ where n is the level of the current multiplicon. This field is left blank for level 2 multiplicons. Since a multiplicon must either have a parent object or an x-segment, both are jointly referred to as the x-object of the multiplicon.

genome_y, list_y For level 2 multiplicons (i.e. containing only 2 segments), these fields describe the second segment (denoted as the y-segment) of a multiplicon. This segment is again identified by the genome it is part of (**genome_y**), the gene list (**list_y**) it is located on.

For multiplicons of level 3 and more, it describes the segment that was last added to the multiplicon using the parent multiplicon (indicated in the parent field) as a profile to detect it.

level Indicates the total number of segments in the multiplicon.

number_of_anchorpoints The number of anchor points (= homologous genes) on the segment last added to a multiplicon, i.e. the y-segment. For level 2 multiplicons, this is the number of homologous gene pairs between segment x and segment y.

profile_length The length of the aligned multiplicon.

begin_x, end_x Begin and end coordinates on the x-object.

begin_y, end_y Begin and end coordinates on the y-segment.

is_redundant Set to true (value -1) or false (value 0). If set to true, this means that a multiplicon is redundant (see above).

3.5.2 The baseclusters table.

This table is mainly for development purposes. When i-ADHoRe detects homologous segments, it does so by first detecting base clusters of anchor points in a gene homology matrix (GHM, see Simillion et al., 2004 for details). These base clusters are first statistically validated and then it is checked if several base clusters are located close enough to each other to be grouped into one metacluster. If a base cluster can not be joined together with another one, a metacluster of a single base cluster is generated. Each metacluster corresponds to a multiplicon in the multiplicons table. Thus, for each multiplicon there will be one or more base clusters. This table is written to the file `baseclusters.txt`.

id An unique identifier for each base cluster.

multiplicon Corresponds to the **id** field of the multiplicons table. Multiple base clusters can be part of one metacluster/multiplicon.

number_of_anchorpoints The number of anchor points (= homologous genes) on the y-segment of the current base cluster.

orientation Indicates if the y segment of the current base cluster has the same (value '+') or opposite (value '-') 5'-3' orientation with respect to the x-segment for level 2 or with respect to the parent multiplicon/profile for higher level multiplicons.

was_twisted Set to true (value -1) or false (value 0). If true, this means that the order of the genes of the y-segment of the current base cluster was reversed before being joined into a metacluster.

cluster_probability A real value indicating the probability that the specified base cluster was generated by chance rather than being generated by true homology. See Simillion et al., 2004 for details of the statistical validation.

3.5.3 The anchorpoints table.

This table describes for each multiplicon all pairs of homologous genes between two segments in the case of level 2 multiplicons or between the parent multiplicon and the segment last added to the multiplicon. This table is written into a file called `anchorpoints.txt`.

id An unique identifier for each anchor point.

multiplicon Refers to the **id** field in the multiplicons table. Indicates the multiplicon the anchorpoint is part of.

basecluster Refers to the **id** field in the base clusters table. Indicates the base cluster the anchor point is part of.

orientation Indicates if both genes involved have the same (value '+') or opposite (value '-') transcriptional orientations.

gene_x The gene on the x-axis in the GHM.

gene_y The gene on the y-axis in the GHM.

coord_x The position of `gene_x` in the x_object of the multiplicon.

coord_y The position of `gene_y` in the y_list of the multiplicon.

real_anchor Set to true (value −1) or false (value 0). If true, it means that this anchor point is counted in the **number_of_anchorpoints** fields in the multiplicons or base clusters table. If false, it is not counted. This occurs when several homologous genes that are aligned on the same position in a profile. If this position shows up as an anchor point on a GHM, it will be counted only once, although each of the genes of that position form a valid homologous pair with the corresponding gene on the y-segment. Again, mainly for development purposes.

3.5.4 The segments table.

Lists all the segments of every multiplicon. This table is written to `segments.txt`.

id Primary key. Use the index numbers of the datastructure in `iADHoRe::record_structure`.

multiplicon The multiplicon the segment belongs to.

genome The genome the segment belongs to.

list The gene list the segment belongs to.

first The first gene (i.e. with the lowest coordinate) of the gene list the segment is part of. This is not always the first gene of the segment, since the order of genes in a multiplicon segment is not always the same as the order on the original gene list. This is because during the profile alignment procedure, the order of genes of the segment or part of it is sometimes reversed.

last The last gene (i.e. with the highest coordinate) of the gene list the segment is part of. Analogous to the **first** field.

order The order in which the segment was added to the multiplicon. E.g. for a level 5 multiplicon, the first segment has order 0 and the last one added has 4.

3.5.5 The genes table.

Lists the position of all genes in all gene lists specified in the settings file. Also tells which genes are marked as tandem repeats. The table is stored in the file `genes.txt`.

id An unique identifier for each gene. The same identifier is used as in the gene list files. Therefore, make sure no two genes, even not in different files, have the same identifier.

genome The genome the gene is located in.

list The gene list of the gene.

coordinate The position of the gene in its gene list.

orientation The transcriptional orientation of the gene, '`+`' or '`-`'.

remapped_coordinate The coordinate of the gene in the remapped gene list (i.e. the gene list from which tandem genes have been removed).

is_tandem Set to true (value -1) or false (value 0). If true, indicates that this gene is part of a tandem array of genes. In i-ADHoRe, a gene is considered a tandem repeat if it is within t_g positions of a homolog on the same gene list. If specified in the settings file t_g is the `tandem_gap` parameter and is otherwise $\frac{g}{2}$ where g is the gap size as specified in the settings file.

is_tandem_representative Set to true (value -1) or false (value 0). In i-ADHoRe all genes that are part of a tandem array are mapped onto the first gene. Thus, this first gene then 'represents' all its siblings in the array. If this field is set to true, the current gene is such a representative.

tandem_representative Refers to the **id** field of this table. Indicates for tandem gene, that gene that functions as its representative (see above).

remapped True if a gene has been remapped, i.e. if it has been removed from the remapped gene list.

3.5.6 The list_elements table.

Lists all positions of genes in multiplicon segments. Again, be aware that the order and orientation of the genes in multiplicon segments is not always the same as that on the original gene list. Also note that the positions of genes in multiplicon segments will not always be consecutive because of gap positions. The table is stored in the file `list_elements.txt`.

id An unique identifier for each element.

segment The segment this element belongs to.

gene The gene occurring on the specific position of the segment. Refers to the `id` field of the genes table.

position The position of the gene in the segment.

orientation The orientation of the gene in the segment.

3.5.7 The clouds table

This table lists all synthenic clouds found in the search which have passed the statistical test (filter). (file is not generated in collinear mode)

id A unique identifier for each synthenic cloud.

genome The genome associated with the cloud (x and y).

list The genelist of the cloud (x and y).

number_of_anchorpoints Number of anchorpoints in the synthenic cloud.

cloud_density Number of anchorpoints divided by the longest side of the bounding box.

dim Horizontal and vertical dimensions of the bounding box of the cloud.

3.5.8 The cloud anchorpoint table

CloudID A unique identifier for each synthenic cloud.

gene The name of the gene associated with the anchorpoint (x and y)

coord The coordinate of the anchorpoint in the dot matrix (x and y)

4 Post processing: i-Visualize

The post processing features of i-ADHoRe include the visualization of aligned multiplicons and dot matrices. To visualize a specific multiplicon an extra executable i-visualize is added. It takes two arguments: the .ini file and the multiplicon id to be visualized. Note that the method uses the output files from i-adhore so these should be available and not be modified. Also note that the color picking is random, so two runs of `i-visualize` will have different colors for the boxes.

```
> i-visualize CONFIGURATIONFILE id
```

With id an integer number indicating the multiplicon id to be visualized.

4.1 Example of AlignmentMultiplion.svg

In figure 1 an example of an aligned multiplicon of level 4 (number of segments) is shown. We see three different types of boxes:

- White boxes represent gaps introduced by the aligner
- Black boxes represent genes which have no homologous links with other genes in the alignment.
- Coloured boxes do have a link with other genes. These are usually aligned on top of each other. It is also possible to find boxes with the same color in the same segment. They are tandem duplicates with a distance larger than the tandem_gap.
- On the left side of the sequences the genome and listnames are written.
- Above every box the gene name is given.

Further on lines are drawn between homologs which haven't been aligned by the algorithm.

4.2 Examples of GHM.png

Dot matrices are visualized with the parameter `visualizeGHM` or `visGPairs`. These can be set in the input file and are described in the section on parameter settings. The first one visualizes all dot matrices, the second one only the ones specified by the user.
Here we see a typical dot matrix between two chromosomes in Arabidopsis thaliana. The dot matrix was generated in the collinear mode. We list all the different features found in the figure:

- White dots: represent homologs between the two genelists which aren't part of any multiplicon or cloud which passed the statistical test.

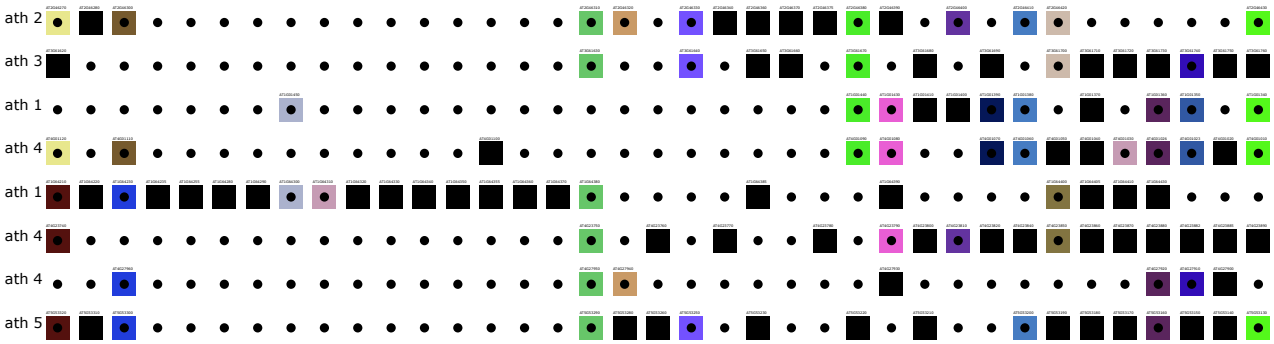

Figure 1: Example of a level-8 Multiplicon after Greedy Graph Alignment. This is multiplicon 7 in DatasetI

- Yellow dots: represent homologs between the two genelists which are part of a multiplicon or cloud which passed the statistical test.
- Red boxes: are the bounding boxes of multiplicons or clouds which haven't passed the statistical
- Green boxes: are the bounding boxes of multiplicons or clouds which are statistically relevant
- Blue dots: represent the confidence interval of the linear regression performed on a multiplicon. (not applicable for cloud mode)

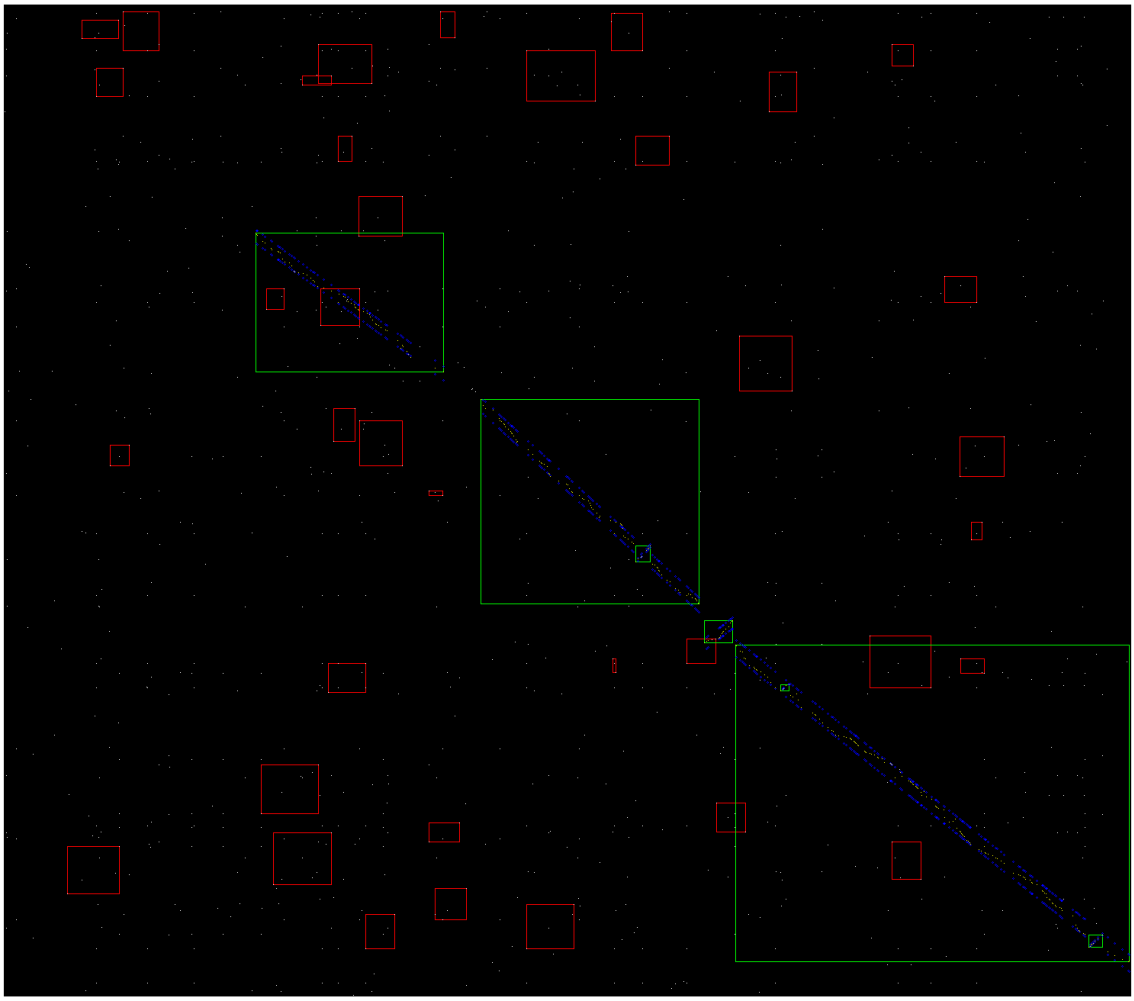

Figure 2: Example of a Gene Homology Matrix (dot matrix) between two chromosomes in *Arabidopsis Thaliana*