Ubiquitous Repeat Tagger and Contaminant Detector Module

Hui-Hsien Chou and Ian Dew

1. Overview

The purpose of the Ubiquitous Repeat Tagger and Contaminant Detector module (AS_URT) is to identify and tag portions of fragments that contain known contaminant and repeat sequences. Fragments containing contaminant sequences must be removed from the processing stream to avoid corrupting intermediate and final assembly results. Regions of fragments containing repeats must be tagged to simplify the jobs and reduce the processing times of the overlapper and unitig graph builder. AS_URT will perform contaminant detection as part of the pre-assembly process, under the control of the Data Management System (DMS), and repeat tagging as part of the assembly process.

2. Requirements

A. Inputs

Two kinds of input data will be processed by the AS_URT module. These will be fragment messages (IFGs) – specifically reads - and screen item messages (ISNs). A limited set of other message types (IDTs, ILKs, IJNs, and RPTs) may be present in the input stream. The specification for these message types is located in the prototyping I/O conventions document. Messages other than IFGs and ISNs will be passed through the AS_URT module with no processing performed on them. Messages will be delivered in the form of files, and one file will be processed per execution of the AS_URT module.

A large (approximately 10Mbp) set of ISN messages is anticipated initially, followed by a steady stream of approximately 200K IFG messages, each about 500bp long, per day. However, new ISN messages may be delivered at any point during a sequencing project, so the AS_URT module must handle all possible mixtures of IFG and ISN messages in a single file. The AS_URT module will create, maintain, and use a screening library that will persist through the duration of a sequencing project. Since the module will be run separately to detect contaminants and repeats, a different library will be created and used for each of these two screen item types. (Currently, the design supports only addition of new screen items. It should support modification and deletion as well.)

All read data will have approximately the same low error rate. Each screen item will specify a unique tolerance or variation limit that defines an alignment match. Presumably contaminants will have low tolerances and repeats will have slightly higher tolerances.

Command line arguments will supply the AS_URT module with names of input and output filenames.

B. Processing

The AS_URT module will have 12 hours on one EV6 processor to screen 200K IFG messages against contaminants and repeats in two separate executions. Within this time constraint, as many contaminants as possible should be detected and as many repeats as possible that would trigger overlaps in the overlapper should be detected. The overlapper will run with a combined error threshold of 6%. Minimizing false negatives takes priority over minimizing false positives.

The high-quality clear range of each read will be screened, and all contaminants and repeats within that region will be tagged. Thus, if more than one repeat are detected in a fragment, all will be tagged.

C. Outputs

There will be two options for AS_URT outputs. The standard form, used for repeats, consists of screened fragment messages (SFGs) and screen match messages (SMAs). SFGs are IFGs with an additional field listing SMAs. Each SMA identifies a match between a fragment and screen item. All other input message types will be included in the output file without modification. The alternate form, used for contaminants, consists of a series of return-delimited records of space-delimited fields, where each record associates a fragment with a matching screen item. Both forms of output will be written to a file.

3. Design

A. Overview

The screener algorithm consists of several steps to find sufficiently similar sub-sequences between screen item and fragment sequences. It first creates a data structure containing all k-mers (k=12 for the Alpha architecture) from a large set of fragment sequences and streams screen item sequence k-mers against it to find identical k-mer hits. Two methods are used for streaming screen items against the set of fragment k-mers. The first uses zero neighborhoods (identical k-mers). The second method uses one neighborhoods (k-mers with zero or one difference). Screen items are processed one at a time, in forward and reverse directions, one k-mer at a time. By design, hits are accumulated in position order (offset into fragment sequence) sorted by Edit Graph diagonal, and sets of hits are considered over an appropriately sized moving window of contiguous diagonals and localized by appropriately sized block (overlapping fragment interval). If the aggregate number of matched bases is sufficient within some subset of diagonals/blocks within a window, a banded local alignment is performed on the fragment/screen item pair around the hits to determine if the sub-sequences are long and similar enough (a match). Each screen item specifies the minimum length required, $\lambda \geq 40$, and maximum variation allowed, $\nu \in [0,.35)$, for a match. In addition, there is an assumed sequencing accuracy, α, currently set to 0.98.

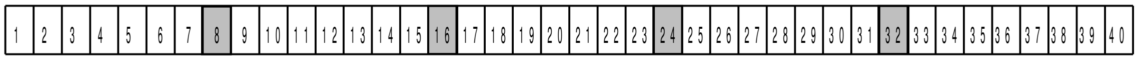
Three computations are critical in determining the effectiveness and speed of this algorithm. First, the k-mer size, κ, and neighborhood size, $\eta \in \{0,1\}$, must be selected for each screen item given α and its values for λ and ν. Next, an array of minimum number of matching k-mer bases, β_i, over a subset of diagonals in the window must be determined. Finally, a minimum local alignment score, γ, must be set to determine whether or not a local alignment qualifies as a match.

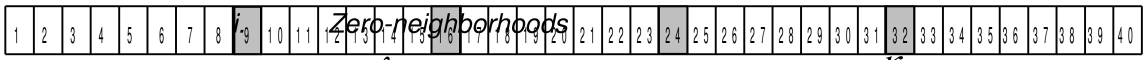
The idea is to examine the number of matched bases and the interval within which they occur over one or more diagonals to determine whether or not to perform a banded local alignment. The goal is to set values for κ and βs that minimize the probability of performing an alignment due to random matches. The discussion below stresses selection of the largest value of κ. This may not be the same as minimizing the probability. Still, the mechanics of the system are independent of the actual values for κ and βs.

B. Selecting κ and β_i thresholds

The maximum number of acceptable base errors, ε, over a length λ is defined by

$$\varepsilon = \lceil \lambda \cdot (1-(\alpha - \nu)) \rceil \tag{1}$$





i. Zero-neighborhoods

For a given λ and ε, the largest acceptable value for κ_0 must be small enough to detect all matching bases when mismatches are spaced evenly over λ. With even spacing, there will be $\varepsilon+1$ hits, and $\lambda-\varepsilon$ bases will be matched. If the mismatches are either all insertions or all deletions, the k-mer hits will be distributed over $\varepsilon+1$ contiguous diagonals. The value of κ_0 is thus

$$\kappa_0 = \left\lfloor \frac{(\lambda-\varepsilon)}{(\varepsilon+1)} \right\rfloor \tag{5}$$

As an example, for $\lambda = 40$ and $\varepsilon = 4$, $\kappa_0 = 7$, and 36 bases are matched (shaded blocks represent mismatches):

If the mismatches are unevenly distributed, hits will still be found using the same κ_0, but fewer matching bases will be covered, and these will be found over fewer, possibly non-contiguous diagonals in window of $\varepsilon+1$ diagonals. For example, if the left-most mismatch is moved one base to the right, the second k-mer from the left will be missed:

In this example, assuming all insertions or all deletions, hits will be found over 4 non-contiguous diagonals in a 5 diagonal window, and a total of 30 bases will be matched. In general, given λ, ε, κ_0, and a number of diagonals, i, on which hits are found in a window of $\varepsilon+1$ diagonals, the number of matched bases must be at least:

$$\beta_i \geq \lambda - \varepsilon\kappa + (\kappa-1)(i-1) \tag{6}$$

Unfortunately, it's not quite so simple. As an example, consider the case where $\lambda = 40$ and $\nu = 0.03$ - subsequences may be 5% different. For a 40 base subsequence, 2 differences are allowed, and (5) yields $\kappa_0 = 12$. For a 41 base subsequence, however, 3 differences are allowed, and (5) specifies $\kappa_0 = 9$. The former will get all appropriate 40 base matches and miss some 41 base matches. Thus, the minimum length of subsequence at which an additional difference is allowed must be considered. This is

$$\lambda_{\varepsilon+1} = \left\lfloor \frac{\varepsilon}{1-(\alpha-\nu)} \right\rfloor + 1 \tag{7}$$

Combining (5) and (7) leads to

$$\kappa_0 = \min\left(\left\lfloor\frac{\lambda-\varepsilon}{\varepsilon+1}\right\rfloor, \left\lfloor\frac{\left\lfloor\frac{\varepsilon}{1-(\alpha-\nu)}\right\rfloor-\varepsilon}{\varepsilon+2}\right\rfloor\right) \tag{8}$$

From (6) and (8), the following table can be derived for $\lambda = 40$ and $\alpha = .98$.

variation	errors	k-mer	num-diags	diag-1	diag-2	diag-3	diag-4	diag-5	diag-6	diag-7	diag-8
ν	ε	κ	Δ	1	2	3	4	5	6	7	8
0.01	2	12	3	16	27	38					
0.02	2	12	3	15	26	37					
0.03	2	9	3	14	22	30					
0.04	3	9	4	15	23	31	37				
0.05	3	7	4	15	21	27	33				
0.06	4	7	5	16	22	28	34	36			
0.07	4	6	5	15	20	25	30	35			
0.08	4	6	5	11	16	21	26	31			
0.09	5	5	6	16	20	24	28	32	35		
0.10	5	5	6	12	16	20	24	28	32		
0.11	6	4	7	16	19	22	25	28	31	34	
0.12	6	4	7	15	18	21	24	27	30	33	
0.13	6	4	7	13	16	19	22	25	28	31	
0.14	7	4	8	12	15	18	21	24	27	30	33

Note that the number of matched bases in the table must be equaled or exceeded within a 40 base subsequence. If the range within which the matches occur is greater than 40 bases, at least one additional base must be matched. In the case of the one-diagonal criteria, if the number of required matches is matched exactly, they must be contiguous (since this analysis assumes the worst case – that mismatches are all insertions or all deletions).

ii. One neighborhoods

Similar logic can be applied to one-neighborhood k-mers. In this case, pairs of errors can be treated as nearly equivalent to individual errors in the zero-neighborhood analysis.

As a similar example to the one used for zero neighborhoods, for $\lambda = 40$ and $\varepsilon = 4$, the largest acceptable value of κ_1 is 13, and 39 bases must be matched if hits are distributed over 3 diagonals in a 5 diagonal window:

The equation for κ_1 is derived from an inequality. Given a λ-length subsequence, a pair of mismatches must occur every $\kappa_1 + 1$ bases to leave a final $\kappa_1 - 1$ subsequence of matching bases. If there is an odd number of errors, this reduces the number of final matching bases by one. Thus,

$$\lambda - \lfloor \varepsilon/2 \rfloor (\kappa_1 + 1) - \varepsilon\%2 \geq \kappa_1 - 1 \tag{9}$$

leading to

$$\kappa_1 = \left\lfloor \frac{\lambda + 1 - \varepsilon\%2 - \lfloor \varepsilon/2 \rfloor}{1 + \lfloor \varepsilon/2 \rfloor} \right\rfloor \tag{10}$$

As in the discussion for zero-neighborhoods, the position of the next allowable error may require a smaller value for κ_1.

$$\kappa_1 = \min\left(\left\lfloor \frac{\lambda + 1 - \varepsilon\%2 - \lfloor \varepsilon/2 \rfloor}{1 + \lfloor \varepsilon/2 \rfloor} \right\rfloor, \left\lfloor \frac{\left\lfloor \frac{\varepsilon}{1-(\alpha-\nu)} \right\rfloor + 2 - (\varepsilon+1)\%2 - \lfloor (\varepsilon+1)/2 \rfloor}{1 + \lfloor (\varepsilon+1)/2 \rfloor} \right\rfloor \right) \tag{11}$$

The minimum number of matched bases on i diagonals in a window of $\varepsilon + 1$ diagonals is given by

$$\beta_i = \lambda + 1 - \lfloor \varepsilon/2 \rfloor \kappa_1 - \varepsilon\%2 + (\kappa_1 - 1)(i-1) \tag{12}$$

From this equation, the following table can be derived for $\lambda = 40$ and $\alpha = .98$. Note that the number of diagonals on which hits may be found (relevant diagonals) is less than the size of the $\varepsilon + 1$ diagonal window, since single mismatches (and diagonals) can be crossed by the one neighborhood k-mer. *(Note: I need to tweak the numbers in the last diagonal for each row since some rows are computed based on $\lambda_{\varepsilon+1}$ and ε+1 but must still be effective at finding λ-length matches.)*

variation	errors	k-mer	num-diags	relevant	diag-1	diag-2	diag-3	diag-4	diag-5	diag-6	diag-7	diag-8
ν	**ε**	**κ**	**Δ**	diagonals	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0.01	2	20	3	2	21	40						
0.02	2	20	3	2	21	40						
0.03	2	20	3	2	21	40						
0.04	3	16	4	2	20	35						
0.05	3	14	4	2	16	29						
0.06	4	13	5	3	15	27	39					
0.07	4	13	5	3	15	27	39					
0.08	4	13	5	3	15	27	39					
0.09	5	11	6	3	14	24	34					
0.10	5	10	6	3	13	22	31					
0.11	6	9	7	4	14	22	30	38				
0.12	6	9	7	4	14	22	30	38				
0.13	6	9	7	4	14	22	30	38				
0.14	7	8	8	4	13	20	27	34				
0.15	7	7	8	4	15	21	27	33				
0.16	8	7	9	5	13	19	25	31	37			
0.17	8	7	9	5	15	21	27	33	39			
0.18	8	7	9	5	13	19	25	31	37			
0.19	9	6	10	5	14	19	24	29	34			
0.20	9	6	10	5	12	17	22	27	32			
0.21	10	6	11	6	14	19	24	29	34	39		
0.22	10	6	11	6	12	17	22	27	32	37		
0.23	10	6	11	6	11	16	21	26	31	36		
0.24	11	5	12	6	14	18	22	26	30	34		

0.25	11	5	12	6	12	16	20	24	28	32		
0.26	12	5	13	7	13	17	21	25	29	33	37	
0.27	12	5	13	7	12	16	20	24	28	32	36	
0.28	12	5	13	7	11	15	19	23	27	31	35	
0.29	13	4	14	7	15	18	21	24	27	30	33	
0.30	13	4	14	7	14	17	20	23	26	29	32	
0.31	14	4	15	8	15	18	21	24	27	30	33	36
0.32	14	4	15	8	14	17	20	23	26	29	32	35
0.33	14	4	15	8	13	16	19	22	25	28	31	34

Note that one-neighborhood 4-mers can handle variations up to 0.33, and in other cases, at least 11 matched bases are required. In the case of the one-diagonal criteria, if the number of required matches is matched exactly, they must be contiguous (since this analysis assumes the worst case – that mismatches are all insertions or all deletions).

iii. Local alignments: computing γ

For a local alignment with a scoring scheme δ such that

$$\delta(a,b)=\begin{cases}\varsigma & if\, a=b\neq -\\ \xi & if\ a\neq b\end{cases} \tag{13}$$

where $\varsigma > 0$, $\xi < 0$, and $|\xi| > \varsigma$, along with a minimum length for a match, λ, an allowable variation in the match, $\nu \in [0,1)$, and an expected sequencing accuracy, α (currently 0.98), the minimum acceptable local alignment score between two sequences is

$$\gamma = \varsigma \cdot (\lambda - \varepsilon) + \xi \cdot \varepsilon \tag{14}$$

The problem with (13) is that short, nearly error-free alignments may generate scores greater than or equal to γ but not be legitimate matches. The length of the local alignment must also be taken into consideration, so in addition to (13), the length of the alignment must be greater than or equal to $\lambda - \varepsilon$.

iv. Implementation

Fragment k-mers and sequence are populated by functions in `AS_URT_frag_data.c`, using the `FragmentData` structure defined in `AS_URT_frag_data.h`. Fragments are processed in groups (10,000 on Intels, 75,000 on Alphas) to reduce memory consumption.

Screen items and the screen item library are defined/managed in the `AS_URT_screen_objects.h/c` and `AS_URT_screen_lib.h/c` files. Key processing parameters are set for each screen item, including k-mer size, number of diagonals, and matching base thresholds in the `SetScreenObjectParameters` function in `AS_URT_screen_objects.c`.

Streaming of screen items against the fragment k-mers is performed by functions in `AS_URT_streaming.c`. Functions therein collect k-mer hits using the `hit_collector` data structure defined/managed in `AS_URT_hit_collector.h/c` which consist of `HitObject`s, defined/managed in `AS_URT_hit_objects.h/c` and consolidate/analyze them using the `hit_analyzer` structure defined/managed in `AS_URT_hit_analyzer.h/c`.

The `hit_collector` collects hits over a moving window of `AS_READ_MAX_LEN` diagonals. As each diagonal is completed, it is added to the `hit_analyzer`, which consolidates the hits and analyzes a smaller moving window of blocks (overlapping intervals within each fragment). The `DetectNewMatches` function in `AS_URT_streaming.c` identifies candidate matches based on the pattern of consolidated hits.

Candidate matches are combined, since nearby candidate matches are likely part of the same, longer match. This also reduces the number of alignments to be performed. A banded alignment is then performed on each candidate match to identify precise start and end positions in the screen item and fragment and to determine whether or not the candidate match is a real match. Real matches are then written to the output file as `IntScreenMatch` messages as part of the `ScreenedFragMesg`.

4. Command Line Interface and Contaminant Warning File Format

A. Command Line Interface

The AS_URT module will be a stand-alone command-line application. Assuming the application is named urc_screener, the command-line interface would be as follows:

```
urc_screener [-(a|f)] [-(c|r)] [-(w|P)] [-s] [-p] [-v]
                        <Screen Library Filename> <Input Filename>
        -a    append to library: either -a or -f is required
        -f    force replacement of library
        -c    screen for contaminants: either -c or -r is required
        -r    screen for ubiquitous repeats
        -w    generate warning message output (i.e., not proto-IO)
        -P    generate ASCII proto-IO output (default is binary)
        -s    generate statistics (celagram & summary files)
        -p    don't pre-parse input file (or update the library)
        -v    verbose
Creates/Maintains/Uses screen library in <Screen Library Filename>
Opens <Input Filename> to read .inp input
Writes output to <Input Filename>.urc or, if specified
        writes warning messages to <Input Filename>.wrn
```

The output file will have the same name prefix as the input file. If the warning file option is specified, the suffix will be ".wrn". Otherwise it will be ".urc".

B. Warning Record Format

Warning records are an alternate output type tailored to meet the needs of DMS and its pre-assembly function of performing contaminant screening. Specifying the –w option on the command line will produce a file with the given filename that contains this form of output.

Each line (record) in the file identifies a match between one fragment and one screen item. Each record contains three space-delimited ASCII fields listing, in order, the fragment's external accession number (uint64) , the screen item's external accession number (uint64), and a number (float: (0,1]) characterizing the fraction of the fragment that is contaminated.

This format should be extended at some point to indicate the portion of the fragment that contains the screen item. This additional information would be useful if, during assembly, it appears that part of the genome is similar to a contaminant.

AUTHORS

Hui-Hsien Chou

Ian Dew

Created: December 2, '98

Revisions:

February 8, '99, Ian Dew

March 12, '99, Ian Dew	Changed ISCs to ISNs and modified .wrn output format.
April 5, '99, Ian Dew	Added one-neighborhood & alignment text.
August 23, '99, Ian Dew	Updated for new screener & added large design section.
February 10, '00, Ian Dew	Updated command line interface for –s and –p options.