Giorgio Gonnella

RGFATools library - API documentation

Version 1.0.1

Table of Contents

Documentation by YARD 0.8.7.6

The Graphical Fragment Assembly (GFA) is a proposed format which allow to describe the product of sequence assembly and is implemented in the GFA class defined in the gfa gem. This gem represents an extension of the GFA class.

Methods in this gem allow, e.g., to randomly orient a segment which has the same connections on both sides, to compute copy numbers and multiply or delete segments according to them, to distribute the links of copies after multipling a segment, or to eliminate edges in the graph which are incompatible with an hamiltonian path. Thereby additional conventions are required, with respect to the GFA specification, which are compatible with it.

Custom optional fields are defined, such as “cn” for the copy number of a segment, “or” for the original segment(s) of a duplicated or merged segment, “mp” for the starting position of original segments in a merged segment, “rp” for the position of possible inversions due to arbitrary orientation of some segments by the program.

Furthermore a convention for the naming of the segments is introduced, which gives a special meaning to the characters “_^()”.

This gem is depends on the “gfa” gem.

Top Level Namespace

Defined Under Namespace

Classes: RGFA

Constant Summary

RGFATools =

```
Module.new
```

Module: RGFATools

Includes:	Edit, Traverse
Included in:	RGFA
Defined in:	lib/rgfatools.rb

Overview

Module defining additional methods for the RGFA class. In the main file is only the method redefinition infrastructure (private methods). The public methods are in the included modules.

Defined Under Namespace

Modules: Edit, Traverse

Constant Summary

Constant Summary

Constants included from Edit

`Edit::LINKS_DISTRIBUTION_POLICY`

Method Summary

Methods included from Traverse

`#merge_linear_path`, `#merge_linear_paths`, `#remove_p_bubble`, `#remove_p_bubbles`

Methods included from Edit

`#apply_copy_number`, `#apply_copy_numbers`, `#compute_copy_numbers`, `#delete_low_coverage_segments`, `#enforce_all_mandatory_links`, `#enforce_segment_mandatory_links`, `#multiply_with_rgfatools`, `#multiply_without_rgfatools`, `#randomly_orient_invertible`, `#randomly_orient_invertibles`, `#remove_dead_ends`, `#remove_self_link`, `#remove_self_links`, `#remove_small_components`, `#set_count_unit_length`, `#set_default_count_tag`

Module: RGFATools::Edit

Included in:	RGFATools
Defined in:	lib/rgfatools/edit.rb

Overview

Methods which edit the graph components without traversal

Constant Summary

LINKS_DISTRIBUTION_POLICY =
Allowed values for the links_distribution_policy option

```
[:off, :auto, :equal, :E, :B]
```

Instance Method Summary

(collapse)

```
- (RGFA) apply_copy_number(segment, count_tag: :cn, links_distribution_policy: :auto, copy_names_suffix: :lowcase, origin_tag: :or, conserve_components: true)
```
Applies the computed copy number to a segment.

```
- (RGFA) apply_copy_numbers(count_tag: :cn, links_distribution_policy: :auto, copy_names_suffix: :lowcase, origin_tag: :or, conserve_components: true)
```
Applies the computed copy number to all segments.

```
- (RGFA) compute_copy_numbers(single_copy_coverage, mincov: single_copy_coverage * 0.25, count_tag: @default[:count_tag], cn_tag: :cn, unit_length: @default[:unit_length])
```
Self.

```
- (RGFA) delete_low_coverage_segments(mincov, count_tag: @default[:count_tag], unit_length: @default[:unit_length])
```
Delete segments which have a coverage under a specified value.

```
- (RGFA) enforce_all_mandatory_links(conserve_components: true)
```
Remove superfluous links in the presence of mandatory links in the entire graph.

```
- (RGFA) enforce_segment_mandatory_links(segment, conserve_components: true)
```
Remove superfluous links in the presence of mandatory links for a single segment.

```
- (RGFA) multiply(segment, factor, copy_names::lowcase, links_distribution_policy::auto, conserve_components:true, origin_tag::or)
```
Create multiple copies of a segment.

```
- (RGFA) multiply_without_rgfatools(segment, factor, copy_names::lowcase, conserve_components:true)
```
Original multiply method of RGFA.

```
- (RGFA) randomly_orient_invertible(segment)
```
Selects a random orientation for an invertible segment.

```
- (RGFA) randomly_orient_invertibles
```
Selects a random orientation for all invertible segments.

`- (RGFA) remove_dead_ends(minlen)`

Remove dead end segments, whose sequence length is under a specified value.

`- (RGFA) remove_self_link(segment)`

Remove links of segment to itself.

`- (RGFA) remove_self_links`

Remove all links of segments to themselves.

`- (RGFA) remove_small_components(minlen)`

Remove connected components whose sum of lengths of the segments is under a specified value.

`- (RGFA) set_count_unit_length(unit_length)`

Sets the unit length (k-mer size, average read lenght or average fragment length) to use for coverage computation (defaults to: 1).

`- (RGFA) set_default_count_tag(tag)`

Sets the count tag to use as default by coverage computations (defaults to: `:RC`).

Instance Method Details

```
- (RGFA) apply_copy_number(segment, count_tag: :cn,
links_distribution_policy: :auto, copy_names_suffix: :lowcase, origin_tag:
:or, conserve_components: true)
```

Applies the computed copy number to a segment

Links distribution policy

Depending on the value of the option `links_ditribution_policy`, an end is eventually selected for distribution of the links.

- `:off`: no distribution performed
- `:E`: links of the E end are distributed
- `:B`: links of the B end are distributed
- `:equal`: select an end for which the number of links is equal to `factor`, if any; if both, then the E end is selected
- `:auto`: automatically select E or B, trying to maximize the number of links which can be deleted

Automatic computation of the copy names:

- First, itis checked if the name of the original segment ends with a relevant string, i.e. a lower case letter (for `:lowcase`), an upper case letter (for `:upcase`), a digit (for `:number`), or the string "`_copy`" plus one or more optional digits (for `:copy`).
- If so, it is assumed, it was already a copy, and it is not altered.
- If not, then `a` (for `:lowcase`), `A` (for `:upcase`), `1` (for `:number`), `_copy` (for `:copy`) is appended to the string.
- Then, in all cases, next (*) is called on the string, until a valid, non-existant name is found for each of the segment copies
- (*) = except for `:copy`, where for the first copy no digit is present, but for the following is, i.e. the segment names will be `:copy`, `:copy2`, `:copy3`, etc.

Parameters:

- **links_distribution_policy** (`RGFATools::Edit::LINKS_DISTRIBUTION_POLICY`) — *(Defaults to: `:auto`)* Determines if and for which end of the segment, links are distributed among the copies. See "Links distribution policy".
- **count_tag** (`Symbol`) — *(defaults to: `:RC` or the value set by #set_default_count_tag)* the count tag to use for coverage computation
- **origin_tag** (`Symbol`) — *(Defaults to: `:or`)* Name of the custom tag to use for storing origin

information.

- **`conserve_components`** (`Boolean`) — *(Defaults to: `true`)* If factor == 0 (i.e. deletion), delete segment only if #cut_segment?(segment) is `false` (see RGFA API).
- **`copy_names_suffix`** (`:lowcase`, `:upcase`, `:number`, `:copy`) — *(Defaults to: `:lowcase`)* Symbol representing a system to compute the names from the name of the original segment. See "Automatic computation of the copy names".
- **`segment`** (`String`, `RGFA::Line::Segment`) — segment name or instance

Returns:

- (`RGFA`) — self

```
- (RGFA) apply_copy_numbers(count_tag: :cn, links_distribution_policy: :auto,
copy_names_suffix: :lowcase, origin_tag: :or, conserve_components: true)
```

Applies the computed copy number to all segments

Links distribution policy

Depending on the value of the option `links_ditribution_policy`, an end is eventually selected for distribution of the links.

- `:off`: no distribution performed
- `:E`: links of the E end are distributed
- `:B`: links of the B end are distributed
- `:equal`: select an end for which the number of links is equal to `factor`, if any; if both, then the E end is selected
- `:auto`: automatically select E or B, trying to maximize the number of links which can be deleted

Automatic computation of the copy names:

- First, itis checked if the name of the original segment ends with a relevant string, i.e. a lower case letter (for `:lowcase`), an upper case letter (for `:upcase`), a digit (for `:number`), or the string "_copy" plus one or more optional digits (for `:copy`).
- If so, it is assumed, it was already a copy, and it is not altered.
- If not, then `a` (for `:lowcase`), `A` (for `:upcase`), `1` (for `:number`), `_copy` (for `:copy`) is appended to the string.
- Then, in all cases, next (*) is called on the string, until a valid, non-existant name is found for each of the segment copies
- (*) = except for `:copy`, where for the first copy no digit is present, but for the following is, i.e. the segment names will be `:copy`, `:copy2`, `:copy3`, etc.

Parameters:

- **`links_distribution_policy`** (`RGFATools::Edit::LINKS_DISTRIBUTION_POLICY`) — *(Defaults to: `:auto`)* Determines if and for which end of the segment, links are distributed among the copies. See "Links distribution policy".
- **`count_tag`** (`Symbol`) — *(defaults to: `:RC` or the value set by #set_default_count_tag)* the count tag to use for coverage computation
- **`origin_tag`** (`Symbol`) — *(Defaults to: `:or`)* Name of the custom tag to use for storing origin information.
- **`conserve_components`** (`Boolean`) — *(Defaults to: `true`)* If factor == 0 (i.e. deletion), delete segment only if #cut_segment?(segment) is `false` (see RGFA API).
- **`copy_names_suffix`** (`:lowcase`, `:upcase`, `:number`, `:copy`) — *(Defaults to: `:lowcase`)* Symbol representing a system to compute the names from the name of the original segment. See "Automatic computation of the copy names".

Returns:

- (`RGFA`) — self

```
- (RGFA) compute_copy_numbers(single_copy_coverage, mincov:
single_copy_coverage * 0.25, count_tag: @default[:count_tag], cn_tag: :cn,
unit_length: @default[:unit_length])
```

Returns self

Parameters:

- **`mincov`** (`Integer`) — *(defaults to: 1/4 of `single_copy_coverage`)* the minimum coverage, cn for segments under this value is set to 0
- **`single_copy_coverage`** (`Integer`) — the coverage that shall be considered to be single copy
- **`cn_tag`** (`Symbol`) — *(defaults to: `:cn`)* the tag to use for storing the copy number
- **`count_tag`** (`Symbol`) — *(defaults to: `:RC` or the value set by #set_default_count_tag)* the count tag to use for coverage computation
- **`unit_length`** (`Integer`) — *(defaults to: 1 or the value set by #set_count_unit_length)* the unit length to use for coverage computation

Returns:

- (`RGFA`) — self

```
- (RGFA) delete_low_coverage_segments(mincov, count_tag:
@default[:count_tag], unit_length: @default[:unit_length])
```

Delete segments which have a coverage under a specified value.

Parameters:

- **`mincov`** (`Integer`) — the minimum coverage
- **`count_tag`** (`Symbol`) — *(defaults to: `:RC` or the value set by #set_default_count_tag)* the count tag to use for coverage computation
- **`unit_length`** (`Integer`) — *(defaults to: 1 or the value set by #set_count_unit_length)* the unit length to use for coverage computation

Returns:

- (`RGFA`) — self

```
- (RGFA) enforce_all_mandatory_links(conserve_components: true)
```

Remove superfluous links in the presence of mandatory links in the entire graph

Parameters:

- **`conserve_components`** (`Boolean`) — *(Defaults to: `true`)* delete links only if #cut_link?(link) is `false` (see RGFA API).

Returns:

- (`RGFA`) — self

```
- (RGFA) enforce_segment_mandatory_links(segment, conserve_components: true)
```

Remove superfluous links in the presence of mandatory links for a single segment

Parameters:

- **`segment`** (`String`, `RGFA::Line::Segment`) — segment name or instance

- **conserve_components** (Boolean) — *(Defaults to: `true`)* delete links only if #cut_link?(link) is `false` (see RGFA API).

Returns:

- (RGFA) — self

```
- (RGFA) multiply(segment, factor, copy_names::lowcase,
links_distribution_policy::auto, conserve_components:true, origin_tag::or)
```

Create multiple copies of a segment.

Complements the multiply method of gfatools with additional functionality. To call the original method use `#multiply_without_rgfatools`.

Automatic computation of the copy names:

- First, itis checked if the name of the original segment ends with a relevant string, i.e. a lower case letter (for `:lowcase`), an upper case letter (for `:upcase`), a digit (for `:number`), or the string "`_copy`" plus one or more optional digits (for `:copy`).
- If so, it is assumed, it was already a copy, and it is not altered.
- If not, then `a` (for `:lowcase`), `A` (for `:upcase`), `1` (for `:number`), `_copy` (for `:copy`) is appended to the string.
- Then, in all cases, next (*) is called on the string, until a valid, non-existant name is found for each of the segment copies
- (*) = except for `:copy`, where for the first copy no digit is present, but for the following is, i.e. the segment names will be `:copy`, `:copy2`, `:copy3`, etc.
- Can be overridden, by providing an array of copy names.

Links distribution policy

Depending on the value of the option `links_ditribution_policy`, an end is eventually selected for distribution of the links.

- `:off`: no distribution performed
- `:E`: links of the E end are distributed
- `:B`: links of the B end are distributed
- `:equal`: select an end for which the number of links is equal to `factor`, if any; if both, then the E end is selected
- `:auto`: automatically select E or B, trying to maximize the number of links which can be deleted

Parameters:

- **factor** (Integer) — multiplication factor; if 0, delete the segment; if 1; do nothing; if > 1; number of copies to create
- **segment** (String, RGFA::Line::Segment) — segment name or instance
- **copy_names** (:lowcase, :upcase, :number, :copy, Array<String>) — *(Defaults to: `:lowcase`)* Array of names for the copies of the segment, or a symbol, which defines a system to compute the names from the name of the original segment. See "Automatic computation of the copy names".
- **conserve_components** (Boolean) — *(Defaults to: `true`)* If factor == 0 (i.e. deletion), delete segment only if #cut_segment?(segment) is `false` (see RGFA API).
- **links_distribution_policy** (RGFATools::Edit::LINKS_DISTRIBUTION_POLICY) — *(Defaults to: `:auto`)* Determines if and for which end of the segment, links are distributed among the copies. See "Links distribution policy".
- **origin_tag** (Symbol) — *(Defaults to: `:or`)* Name of the custom tag to use for storing origin information.

Returns:

- (RGFA) — self

```
- (RGFA) multiply_without_rgfatools(segment, factor, copy_names::lowcase,
conserve_components:true)
```

Original multiply method of RGFA. See the RGFA API documentation for detail.

Returns:

- (`RGFA`) — self

```
- (RGFA) randomly_orient_invertible(segment)
```

Selects a random orientation for an invertible segment

Parameters:

- **`segment`** (`String`, `RGFA::Line::Segment`) — segment name or instance

Returns:

- (`RGFA`) — self

```
- (RGFA) randomly_orient_invertibles
```

Selects a random orientation for all invertible segments

Returns:

- (`RGFA`) — self

```
- (RGFA) remove_dead_ends(minlen)
```

Remove dead end segments, whose sequence length is under a specified value.

Parameters:

- **`minlen`** (`Integer`) — the minimum length

Returns:

- (`RGFA`) — self

```
- (RGFA) remove_self_link(segment)
```

Remove links of segment to itself

Parameters:

- **`segment`** (`String`, `RGFA::Line::Segment`) — segment name or instance

Returns:

- (`RGFA`) — self

```
- (RGFA) remove_self_links
```

Remove all links of segments to themselves

Returns:

- (RGFA) — self

`- (RGFA) remove_small_components(minlen)`

Remove connected components whose sum of lengths of the segments is under a specified value.

Parameters:

- **minlen** (Integer) — the minimum length

Returns:

- (RGFA) — self

`- (RGFA) set_count_unit_length(unit_length)`

Sets the unit length (k-mer size, average read lenght or average fragment length) to use for coverage computation *(defaults to: 1)*.

Parameters:

- **unit_length** (Integer) — the unit length to use

Returns:

- (RGFA) — self

`- (RGFA) set_default_count_tag(tag)`

Sets the count tag to use as default by coverage computations *(defaults to: :RC)*.

Parameters:

- **tag** (Symbol) — the tag to use

Returns:

- (RGFA) — self

Module: RGFATools::Traverse

Included in:	RGFATools
Defined in:	lib/rgfatools/traverse.rb

Overview

Methods for the RGFA class, which involve a traversal of the graph following links

Constant Summary

Instance Method Summary

(collapse)

`- (RGFA) merge_linear_path(segpath, **options)`

Merge a linear path, i.e.

`- (RGFA) merge_linear_paths(**options)`

Merge all linear paths in the graph, i.e.

`- (RGFA) remove_p_bubble(segment_end1, segment_end2, count_tag: @default[:count_tag], unit_length: @default[:unit_length])`

Removes a p-bubble between segment_end1 and segment_end2.

`- (RGFA) remove_p_bubbles`

Removes all p-bubbles in the graph.

Instance Method Details

`- (RGFA) merge_linear_path(segpath, **options)`

Merge a linear path, i.e. a path of segments without extra-branches. Extends the RGFA method, with additional functionality:

- `name`: the name of the merged segment is set to the name of the single segments joined by underscore (_). If a name already contained an underscore, it is splitted before merging. Whenever a segment is reversed complemented, its name (or the name of all its components) is suffixed with a ^; if the last letter was already ^, it is removed; if it contained _ the name is splitted, the elements reversed and joined back using _; round parentheses are removed from the name before processing and added back after it.
- `:or`: keeps track of the origin of the merged segment; the origin tag is set to an array of :or or name (if no :or available) tags of the segment which have been merged; the character ^ is assigned the same meaning as in `name`
- `:rn`: tag used to store possible inversion positions and it is updated by this method; i.e. it is passed from the single segments to the merged segment, and the coordinates updated
- `:mp`: tag used to store the position of the single segments in the merged segment; it is created or updated by this method

Limitations: all containments und paths involving merged segments are deleted.

Parameters:

- **`segpath`** (`Array<RGFA::SegmentEnd>`) — a linear path, such as that retrieved by #linear_path (see RGFA API documentation)

- **options** (`Hash`) — optional keyword arguments

Options Hash (`**options`):

- **:merged_name** (`String`, `:short`, `nil`) — default: `nil` — if nil, the merged_name is automatically computed; if :short, a name is computed starting with “merged1” and calling next until an available name is founf; if String, the name to use
- **:cut_counts** (`Boolean`) — default: `false` — if true, total count in merged segment m, composed of segments s of set S is multiplied by the factor Sum(|s in S|)/|m|
- **:disable_tracking** (`Boolean`) — default: `false` — if true, the original #multiply of RGFA without RGFATools is called.

Returns:

- (`RGFA`) — self

See Also:

- #merge_linear_paths

```
- (RGFA) merge_linear_paths(**options)
```

Merge all linear paths in the graph, i.e. paths of segments without extra-branches Extends the RGFA method, with additional functionality:

- `name`: the name of the merged segment is set to the name of the single segments joined by underscore (_). If a name already contained an underscore, it is splitted before merging. Whenever a segment is reversed complemented, its name (or the name of all its components) is suffixed with a ^; if the last letter was already ^, it is removed; if it contained _ the name is splitted, the elements reversed and joined back using _; round parentheses are removed from the name before processing and added back after it.
- `:or`: keeps track of the origin of the merged segment; the origin tag is set to an array of :or or name (if no :or available) tags of the segment which have been merged; the character ^ is assigned the same meaning as in `name`
- `:rn`: tag used to store possible inversion positions and it is updated by this method; i.e. it is passed from the single segments to the merged segment, and the coordinates updated
- `:mp`: tag used to store the position of the single segments in the merged segment; it is created or updated by this method

Limitations: all containments und paths involving merged segments are deleted.

Parameters:

- **options** (`Hash`) — optional keyword arguments

Options Hash (`**options`):

- **:merged_name** (`String`, `:short`, `nil`) — default: `nil` — if nil, the merged_name is automatically computed; if :short, a name is computed starting with “merged1” and calling next until an available name is founf; if String, the name to use
- **:cut_counts** (`Boolean`) — default: `false` — if true, total count in merged segment m, composed of segments s of set S is multiplied by the factor Sum(|s in S|)/|m|
- **:disable_tracking** (`Boolean`) — default: `false` — if true, the original #multiply of RGFA without RGFATools is called.

Returns:

- (`RGFA`) — self

```
- (RGFA) remove_p_bubble(segment_end1, segment_end2, count_tag:
@default[:count_tag], unit_length: @default[:unit_length])
```

Removes a p-bubble between segment_end1 and segment_end2

Parameters:

- **`segment_end1`** (`RGFA::SegmentEnd`) — a segment end
- **`segment_end2`** (`RGFA::SegmentEnd`) — another segment end
- **`count_tag`** (`Symbol`) — *(defaults to: `:RC` or the value set by Edit#set_default_count_tag)* the count tag to use for coverage computation
- **`unit_length`** (`Integer`) — *(defaults to: 1 or the value set by Edit#set_count_unit_length)* the unit length to use for coverage computation

Returns:

- (`RGFA`) — self

- (RGFA) **remove_p_bubbles**

Removes all p-bubbles in the graph

Returns:

- (`RGFA`) — self

Class: RGFA

Inherits:	Object	show all
Includes:	RGFATools	
Defined in:	lib/rgfatools.rb	

Overview

The main class of RRGFA. See the RRGFA API documentation.

Constant Summary

Constant Summary

Constants included from RGFATools::Edit

`RGFATools::Edit::LINKS_DISTRIBUTION_POLICY`

Method Summary

Methods included from RGFATools::Traverse

`#merge_linear_path`, `#merge_linear_paths`, `#remove_p_bubble`, `#remove_p_bubbles`

Methods included from RGFATools::Edit

`#apply_copy_number`, `#apply_copy_numbers`, `#compute_copy_numbers`, `#delete_low_coverage_segments`, `#enforce_all_mandatory_links`, `#enforce_segment_mandatory_links`, `#multiply_with_rgfatools`, `#multiply_without_rgfatools`, `#randomly_orient_invertible`, `#randomly_orient_invertibles`, `#remove_dead_ends`, `#remove_self_link`, `#remove_self_links`, `#remove_small_components`, `#set_count_unit_length`, `#set_default_count_tag`