

---

# **Cython\_GedLib Documentation**

***Release 1.0***

**Natacha Lambert**

**Jul 19, 2019**



## CONTENTS:

<b>1</b>	<b>Python GedLib module</b>	<b>1</b>
1.1	Authors . . . . .	1
1.2	Classes & Functions . . . . .	1
<b>2</b>	<b>How to add your own editCost class</b>	<b>17</b>
2.1	C++ side . . . . .	17
2.2	Python side . . . . .	17
<b>3</b>	<b>Indices and tables</b>	<b>19</b>
	<b>Python Module Index</b>	<b>21</b>
	<b>Index</b>	<b>23</b>



## PYTHON GEDLIB MODULE

This module allow to use a C++ library for edit distance between graphs (GedLib) with Python.

### 1.1 Authors

David Blumenthal Natacha Lambert

Copyright (C) 2019 by all the authors

### 1.2 Classes & Functions

**exception** `PythonGedLib.EditCostError`

Class for Edit Cost Error. Raise an error if an edit cost function doesn't exist in the library (not in `listOfEditCostOptions`).

**Attribute message** The message to print when an error is detected.

**exception** `PythonGedLib.Error`

Class for error's management. This one is general.

**exception** `PythonGedLib.InitError`

Class for Init Error. Raise an error if an init option doesn't exist in the library (not in `listOfInitOptions`).

**Attribute message** The message to print when an error is detected.

**exception** `PythonGedLib.MethodError`

Class for Method Error. Raise an error if a computation method doesn't exist in the library (not in `listOfMethodOptions`).

**Attribute message** The message to print when an error is detected.

`PythonGedLib.PyAddEdge()`

Adds an edge on a graph selected by its ID.

**Parameters**

- **graphID** (*size\_t*) – The ID of the wanted graph
- **tail** (*string*) – The ID of the tail node for the new edge
- **head** (*string*) – The ID of the head node for the new edge
- **edgeLabel** (*dict{string : string}*) – The label of the new edge
- **ignoreDuplicates** (*bool*) – If True, duplicate edges are ignored, otherwise it's raise an error if an existing edge is added. True by default

---

**Note:** You can also use this function after initialization, but only on a newly added graph. Call `PyInitEnv()` after you're finished your modifications.

---

`PythonGedLib.PyAddGraph()`

Adds a empty graph on the environment, with its name and its class. Nodes and edges will be add in a second time.

**Parameters**

- **name** (*string*) – The name of the new graph, an empty string by default
- **classe** (*string*) – The class of the new graph, an empty string by default

**Returns** The ID of the newly graphe

**Return type** `size_t`

---

**Note:** You can call this function without parameters. You can also use this function after initialization, call `PyInitEnv()` after you're finished your modifications.

---

`PythonGedLib.PyAddNode()`

Adds a node on a graph selected by its ID. A ID and a label for the node is required.

**Parameters**

- **graphID** (*size\_t*) – The ID of the wanted graph
- **nodeID** (*string*) – The ID of the new node
- **nodeLabel** (*dict{string : string}*) – The label of the new node

---

**Note:** You can also use this function after initialization, but only on a newly added graph. Call `PyInitEnv()` after you're finished your modifications.

---

`PythonGedLib.PyAddSymmetricalEdge()`

Adds a symmetrical edge on a graph selected by its ID.

**Parameters**

- **graphID** (*size\_t*) – The ID of the wanted graph
- **tail** (*string*) – The ID of the tail node for the new edge
- **head** (*string*) – The ID of the head node for the new edge
- **edgeLabel** (*dict{string : string}*) – The label of the new edge

---

**Note:** You can also use this function after initialization, but only on a newly added graph. Call `PyInitEnv()` after you're finished your modifications.

---

`PythonGedLib.PyClearGraph()`

Deletes a graph, selected by its ID, to the environment.

**Parameters** **graphID** (*size\_t*) – The ID of the wanted graph

---

**Note:** Call `PyInit()` after you're finished your modifications.

---

`PythonGedLib.PyGetAllGraphIds()`

Searchs all the IDs of the loaded graphs in the environment.

**Returns** The list of all graphs's Ids

**Return type** `list[size_t]`

---

**Note:** The last ID is equal to (number of graphs - 1). The order correspond to the loading order.

---

`PythonGedLib.PyGetAllMap()`

Returns a vector which contains the forward and the backward maps between nodes of the two indicated graphs.

**Parameters**

- `g(size_t)` – The Id of the first compared graph
- `h(size_t)` – The Id of the second compared graph

**Returns** The forward and backward maps to the adjacence matrix between nodes of the two graphs

**Return type** `list[list[np_uint32]]`

**See also:**

`PyRunMethod()`, `PyGetUpperBound()`, `PyGetLowerBound()`, `PyGetForwardMap()`, `PyGetBackwardMap()`, `PyGetRuntime()`, `PyQuasimetricCost()`

**Warning:** `PyRunMethod()` between the same two graph must be called before this function.

---

**Note:** This function duplicates data so please don't use it. I also don't know how to connect the two map to reconstruct the adjacence matrix. Please come back when I know how it's work !

---

`PythonGedLib.PyGetAssignmentMatrix()`

Returns the Assignment Matrix between two selected graphs `g` and `h`.

**Parameters**

- `g(size_t)` – The Id of the first compared graph
- `h(size_t)` – The Id of the second compared graph

**Returns** The Assignment Matrix between the two selected graph.

**Return type** `list[list[int]]`

**See also:**

`PyRunMethod()`, `PyGetForwardMap()`, `PyGetBackwardMap()`, `PyGetNodeImage()`, `PyGetNodePreImage()`, `PyGetNodeMap()`

**Warning:** PyRunMethod() between the same two graph must be called before this function.

---

**Note:** This function creates datas so use it if necessary.

---

PythonGedLib.**PyGetBackwardMap**()

Returns the backward map (or the half of the adjacence matrix) between nodes of the two indicated graphs.

**Parameters**

- **g**(size\_t) – The Id of the first compared graph
- **h**(size\_t) – The Id of the second compared graph

**Returns** The backward map to the adjacence matrix between nodes of the two graphs

**Return type** list[np\_uint32]

**See also:**

PyRunMethod(), PyGetUpperBound(), PyGetLowerBound(), PyGetForwardMap(), PyGetRuntime(), PyQuasi-metricCost(), PyGetNodeMap(), PyGetAssignmentMatrix()

**Warning:** PyRunMethod() between the same two graph must be called before this function.

---

**Note:** I don't know how to connect the two map to reconstruct the adjacence matrix. Please come back when I know how it's work !

---

PythonGedLib.**PyGetDummyNode**()

Returns the ID of a dummy node.

**Returns** The ID of the dummy node (18446744073709551614 for my computer, the hugest number possible)

**Return type** size\_t

---

**Note:** A dummy node is used when a node isn't associated to an other node.

---

PythonGedLib.**PyGetEditCostOptions**()

Searchs the differents edit cost functions and returns the result.

**Returns** The list of edit cost functions

**Return type** list[string]

**Warning:** This function is useless for an external use. Please use directly listOfEditCostOptions.

---

**Note:** Prefer the listOfEditCostOptions attribute of this module.

---

PythonGedLib.**PyGetForwardMap**()

Returns the forward map (or the half of the adjacence matrix) between nodes of the two indicated graphs.



#### Parameters

- **g** (*size\_t*) – The Id of the first compared graph
- **h** (*size\_t*) – The Id of the second compared graph

**Returns** The forward map to the adjacence matrix between nodes of the two graphs

**Return type** list[npy\_uint32]

#### See also:

PyRunMethod(), PyGetUpperBound(), PyGetLowerBound(), PyGetBackwardMap(), PyGetRuntime(), PyQuasimetricCost(), PyGetNodeMap(), PyGetAssignmentMatrix()

**Warning:** PyRunMethod() between the same two graph must be called before this function.

---

**Note:** I don't know how to connect the two map to reconstruct the adjacence matrix. Please come back when I know how it's work !

---

PythonGedLib.**PyGetGraphAdjacenceMatrix** ()

Searchs and returns the adjacence list of a graph, selected by its ID.

**Parameters** **graphID** (*size\_t*) – The ID of the wanted graph

**Returns** The adjacence list of the selected graph

**Return type** list[list[*size\_t*]]

---

**Note:** These functions allow to collect all the graph's informations.

---

PythonGedLib.**PyGetGraphClass** ()

Returns the class of a graph with its ID.

**Parameters** **id** (*size\_t*) – The ID of the wanted graph

**Returns** The class of the graph which correpond to the ID

**Return type** *string*

#### See also:

PyGetGraphClass()

---

**Note:** An empty string can be a class.

---

PythonGedLib.**PyGetGraphEdges** ()

Searchs and returns all the edges on a graph, selected by its ID.

**Parameters** **graphID** (*size\_t*) – The ID of the wanted graph

**Returns** The list of edges on the selected graph

**Return type** dict{tuple(*size\_t*,*size\_t*) : dict{string : string}}

---

**Note:** These functions allow to collect all the graph's informations.

---

`PythonGedLib.PyGetGraphIds()`

Searchs the first and last IDs of the loaded graphs in the environment.

**Returns** The pair of the first and the last graphs Ids

**Return type** tuple(size\_t, size\_t)

---

**Note:** Prefer this function if you have huges structures with lots of graphs.

---

`PythonGedLib.PyGetGraphInternalId()`

Searchs and returns the internal Id of a graph, selected by its ID.

**Parameters** `graphID (size_t)` – The ID of the wanted graph

**Returns** The internal ID of the selected graph

**Return type** size\_t

---

**Note:** These functions allow to collect all the graph's informations.

---

`PythonGedLib.PyGetGraphName()`

Returns the name of a graph with its ID.

**Parameters** `id (size_t)` – The ID of the wanted graph

**Returns** The name of the graph which correpond to the ID

**Return type** string

**See also:**

`PyGetGraphClass()`

---

**Note:** An empty string can be a name.

---

`PythonGedLib.PyGetGraphNodeLabels()`

Searchs and returns all the labels of nodes on a graph, selected by its ID.

**Parameters** `graphID (size_t)` – The ID of the wanted graph

**Returns** The list of labels's nodes on the selected graph

**Return type** list[dict{string : string}]

---

**Note:** These functions allow to collect all the graph's informations.

---

`PythonGedLib.PyGetGraphNumEdges()`

Searchs and returns the number of edges on a graph, selected by its ID.

**Parameters** `graphID (size_t)` – The ID of the wanted graph

**Returns** The number of edges on the selected graph

**Return type** size\_t

---

**Note:** These functions allow to collect all the graph's informations.

---

`PythonGedLib.PyGetGraphNumNodes()`

Searchs and returns the number of nodes on a graph, selected by its ID.

**Parameters** `graphID (size_t)` – The ID of the wanted graph

**Returns** The number of nodes on the selected graph

**Return type** `size_t`

---

**Note:** These functions allow to collect all the graph's informations.

---

`PythonGedLib.PyGetInitOptions()`

Searchs the differents initialization parameters for the environment computation for graphs and returns the result.

**Returns** The list of options to initialize the computation environment

**Return type** `list[string]`

**Warning:** This function is useless for an external use. Please use directly `listOfInitOptions`.

---

**Note:** Prefer the `listOfInitOptions` attribute of this module.

---

`PythonGedLib.PyGetInitime()`

Returns the initialization time.

**Returns** The initialization time

**Return type** `double`

`PythonGedLib.PyGetLowerBound()`

Returns the lower bound of the edit distance cost between two graphs `g` and `h`.

**Parameters**

- `g (size_t)` – The Id of the first compared graph
- `h (size_t)` – The Id of the second compared graph

**Returns** The lower bound of the edit distance cost

**Return type** `double`

**See also:**

`PyRunMethod()`, `PyGetUpperBound()`, `PyGetForwardMap()`, `PyGetBackwardMap()`, `PyGetRuntime()`, `PyQuasimetricCost()`

**Warning:** `PyRunMethod()` between the same two graph must be called before this function.

---

**Note:** This function can be ignored, because lower bound doesn't have a crucial utility.

---

`PythonGedLib.PyGetMethodOptions()`

Searchs the differents method for edit distance computation between graphs and returns the result.

**Returns** The list of method to compute the edit distance between graphs

**Return type** list[string]

**Warning:** This function is useless for an external use. Please use directly listOfMethodOptions.

---

**Note:** Prefer the listOfMethodOptions attribute of this module.

---

PythonGedLib.**PyGetNodeImage**()

Returns the node's image in the adjacence matrix, if it exists.

**Parameters**

- **g**(size\_t) – The Id of the first compared graph
- **h**(size\_t) – The Id of the second compared graph
- **nodeID** – The ID of the node which you want to see the image

:type nodeID size\_t :return: The ID of the image node :rtype: size\_t

**See also:**

PyRunMethod(), PyGetForwardMap(), PyGetBackwardMap(), PyGetNodePreImage(), PyGetNodeMap(), PyGetAssignmentMatrix()

**Warning:** PyRunMethod() between the same two graph must be called before this function.

---

**Note:** Use BackwardMap's Node to find its images ! You can also use PyGetForwardMap() and PyGetBackwardMap().

---

PythonGedLib.**PyGetNodeMap**()

Returns the Node Map, like C++ NodeMap.

**Parameters**

- **g**(size\_t) – The Id of the first compared graph
- **h**(size\_t) – The Id of the second compared graph

**Returns** The Node Map between the two selected graph.

**Return type** list[tuple(size\_t, size\_t)]

**See also:**

PyRunMethod(), PyGetForwardMap(), PyGetBackwardMap(), PyGetNodeImage(), PyGetNodePreImage(), PyGetAssignmentMatrix()

**Warning:** PyRunMethod() between the same two graph must be called before this function.

---

**Note:** This function creates datas so use it if necessary, however you can understand how assignement works with this example.

---

PythonGedLib.**PyGetNodePreImage**()

Returns the node's preimage in the adjacence matrix, if it exists.

**Parameters**

- **g**(*size\_t*) – The Id of the first compared graph
- **h**(*size\_t*) – The Id of the second compared graph
- **nodeID** – The ID of the node which you want to see the preimage

:type nodeID *size\_t* :return: The ID of the preimage node :rtype: *size\_t*

**See also:**

PyRunMethod(), PyGetForwardMap(), PyGetBackwardMap(), PyGetNodeImage(), PyGetNodeMap(), PyGetAssignmentMatrix()

**Warning:** PyRunMethod() between the same two graph must be called before this function.

---

**Note:** Use ForwardMap's Node to find its images ! You can also use PyGetForwardMap() and PyGetBackwardMap().

---

PythonGedLib.**PyGetOriginalNodeIds**()

Searchs and returns all th Ids of nodes on a graph, selected by its ID.

**Parameters** **graphID**(*size\_t*) – The ID of the wanted graph

**Returns** The list of IDs's nodes on the selected graph

**Return type** list[string]

---

**Note:** These functions allow to collect all the graph's informations.

---

PythonGedLib.**PyGetRuntime**()

Returns the runtime to compute the edit distance cost between two graphs g and h

**Parameters**

- **g**(*size\_t*) – The Id of the first compared graph
- **h**(*size\_t*) – The Id of the second compared graph

**Returns** The runtime of the computation of edit distance cost between the two selected graphs

**Return type** double

**See also:**

PyRunMethod(), PyGetUpperBound(), PyGetLowerBound(), PyGetForwardMap(), PyGetBackwardMap(), PyQuasimetricCost()

**Warning:** PyRunMethod() between the same two graph must be called before this function.

---

**Note:** Python is a bit longer than C++ due to the functions's encapsulate.

---

PythonGedLib.**PyGetUpperBound**()

Returns the upper bound of the edit distance cost between two graphs g and h.

**Parameters**

- **g** (*size\_t*) – The Id of the first compared graph
- **h** (*size\_t*) – The Id of the second compared graph

**Returns** The upper bound of the edit distance cost

**Return type** double

**See also:**

PyRunMethod(), PyGetLowerBound(), PyGetForwardMap(), PyGetBackwardMap(), PyGetRuntime(), PyQuasimetricCost()

**Warning:** PyRunMethod() between the same two graph must be called before this function.

---

**Note:** The upper bound is equivalent to the result of the pessimist edit distance cost. Methods are heuristics so the library can't compute the real perfect result because it's NP-Hard problem.

---

PythonGedLib.**PyHungarianLSAP**()

Applies the hungarian algorithm (LSAP) on a matrix Cost.

**Parameters** **matrixCost** (*vector[vector[size\_t]]*) – The matrix Cost

**Returns** The values of rho, varrho, u and v, in this order

**Return type** vector[vector[size\_t]]

PythonGedLib.**PyHungarianLSAPE**()

Applies the hungarian algorithm (LSAPE) on a matrix Cost.

**Parameters** **matrixCost** (*vector[vector[double]]*) – The matrix Cost

**Returns** The values of rho, varrho, u and v, in this order

**Return type** vector[vector[double]]

PythonGedLib.**PyInitEnv**()

Initializes the environment with the chosen edit cost function and graphs.

**Parameters** **initOption** (*string*) – The name of the init option, "EA-GER\_WITHOUT\_SHUFFLED\_COPIES" by default

**See also:**

listOfInitOptions

**Warning:** No modification were allowed after initialization. Try to make sure your choices is correct. You can though clear or add a graph, but recall `PyInitEnv()` after that.

---

**Note:** Try to make sure the option exists with `listOfInitOptions` or choose no options, raise an error otherwise.

---

`PythonGedLib.PyInitMethod()`

Init the environment with the set method.

**See also:**

`PySetMethod()`, `listOfMethodOptions`

---

**Note:** Call this function after set the method. You can't launch computation or change the method after that.

---

`PythonGedLib.PyIsInitialized()`

Checks and returns if the computation environment is initialized or not.

**Returns** True if it's initialized, False otherwise

**Return type** `bool`

---

**Note:** This function exists for internals verifications but you can use it for your code.

---

`PythonGedLib.PyLoadGXLGraph()`

Loads some GXL graphs on the environment which is in a same folder, and present in the XMLfile.

**Parameters**

- **pathFolder** (*string*) – The folder's path which contains GXL graphs
- **pathXML** (*string*) – The XML's path which indicates which graphs you want to load

---

**Note:** You can call this function multiple times if you want, but not after an init call.

---

`PythonGedLib.PyQuasimetricCost()`

Checks and returns if the edit costs are quasimetric.

**Parameters**

- **g** (*size\_t*) – The Id of the first compared graph
- **h** (*size\_t*) – The Id of the second compared graph

**Returns** True if it's verified, False otherwise

**Return type** `bool`

**See also:**

`PyRunMethod()`, `PyGetUpperBound()`, `PyGetLowerBound()`, `PyGetForwardMap()`, `PyGetBackwardMap()`, `PyGetRuntime()`

**Warning:** `PyRunMethod()` between the same two graph must be called before this function.

`PythonGedLib.PyRestartEnv()`

Restarts the environment variable. All data related to it will be delete.

**Warning:** This function deletes all graphs, computations and more so make sure you don't need anymore your environment.

---

**Note:** You can now delete and add some graphs after initialization so you can avoid this function.

---

`PythonGedLib.PyRunMethod()`

Computes the edit distance between two graphs *g* and *h*, with the edit cost function and method computation selected.

**Parameters**

- ***g*** (*size\_t*) – The Id of the first graph to compare
- ***h*** (*size\_t*) – The Id of the second graph to compare

**See also:**

`PyGetUpperBound()`, `PyGetLowerBound()`, `PyGetForwardMap()`, `PyGetBackwardMap()`, `PyGetRuntime()`, `PyQuasimetricCost()`

---

**Note:** This function only compute the distance between two graphs, without returning a result. Use the `differeents` function to see the result between the two graphs.

---

`PythonGedLib.PySetEditCost()`

Sets an edit cost function to the environment, if its exists.

**Parameters**

- **`editCost`** (*string*) – The name of the edit cost function
- **`editCostConstant`** (*list*) – The parameters you will add to the `editCost`, empty by default

---

**Note:** Try to make sure the edit cost function exists with `listOfEditCostOptions`, raise an error otherwise.

---

`PythonGedLib.PySetMethod()`

Sets a computation method to the environment, if its exists.

**Parameters**

- **`method`** (*string*) – The name of the computation method
- **`options`** (*string*) – The options of the method (like bash options), an empty string by default

**See also:**

`PyInitMethod()`, `listOfMethodOptions`

---

**Note:** Try to make sure the edit cost function exists with `listOfMethodOptions`, raise an error otherwise. Call `PyInitMethod()` after your set.

---



`PythonGedLib.PySetPersonalEditCost()`

Sets an personal edit cost function to the environment.

**Parameters** `editCostConstant` (*list*) – The parameters you will add to the editCost, empty by default

`PythonGedLib.addNxGraph()`

Add a Graph (made by networkx) on the environment. Be careful to respect the same format as GXL graphs for labelling nodes and edges.

**Parameters**

- **g** (*networkx.graph*) – The graph to add (networkx graph)
- **ignoreDuplicates** (*bool*) – If True, duplicate edges are ignored, otherwise it's raise an error if an existing edge is added. True by default

**Returns** The ID of the newly added graphe

**Return type** `size_t`

---

**Note:** The NX graph must respect the GXL structure. Please see how a GXL graph is construct.

---

`PythonGedLib.addRandomGraph()`

Add a Graph (not GXL) on the environment. Be careful to respect the same format as GXL graphs for labelling nodes and edges.

**Parameters**

- **name** (*string*) – The name of the graph to add, can be an empty string
- **classe** (*string*) – The classe of the graph to add, can be an empty string
- **listOfNodes** (*list[tuple(size\_t, dict{string : string})]*) – The list of nodes to add
- **listOfEdges** (*list[tuple(tuple(size\_t, size\_t), dict{string : string})]*) – The list of edges to add
- **ignoreDuplicates** (*bool*) – If True, duplicate edges are ignored, otherwise it's raise an error if an existing edge is added. True by default

**Returns** The ID of the newly added graphe

**Return type** `size_t`

---

**Note:** The graph must respect the GXL structure. Please see how a GXL graph is construct.

---

`PythonGedLib.appel()`

Calls an example only in C++. Nothing usefull, that's why you must ignore this function.

`PythonGedLib.computeEditDistanceOnGXLGraphs()`

Computes all the edit distance between each GXL graphs on the folder and the XML file.

**Parameters**

- **pathFolder** (*string*) – The folder's path which contains GXL graphs
- **pathXML** (*string*) – The XML's path which indicates which graphes you want to load
- **editCost** (*string*) – The name of the edit cost function

- **method** (*string*) – The name of the computation method
- **options** (*string*) – The options of the method (like bash options), an empty string by default
- **initOption** (*string*) – The name of the init option, “EAGER\_WITHOUT\_SHUFFLED\_COPIES” by default

**Returns** The list of the first and last-1 ID of graphs

**Return type** tuple(size\_t, size\_t)

**See also:**

listOfEditCostOptions, listOfMethodOptions, listOfInitOptions

---

**Note:** Make sure each parameter exists with your architecture and these lists : listOfEditCostOptions, listOfMethodOptions, listOfInitOptions.

---

PythonGedLib.**computeEditDistanceOnNxGraphs** ()

Computes all the edit distance between each NX graphs on the dataset.

**Parameters**

- **dataset** (*list[networksx.graph]*) – The list of graphs to add and compute
- **classes** (*string*) – The classe of all the graph, can be an empty string
- **editCost** (*string*) – The name of the edit cost function
- **method** (*string*) – The name of the computation method
- **options** (*string*) – The options of the method (like bash options), an empty string by default
- **initOption** (*string*) – The name of the init option, “EAGER\_WITHOUT\_SHUFFLED\_COPIES” by default

**Returns** Two matrix, the first with edit distances between graphs and the second the nodeMap between graphs. The result between g and h is one the [g][h] coordinates.

**Return type** list[list[double]], list[list[list[tuple(size\_t, size\_t)]]]

**See also:**

listOfEditCostOptions, listOfMethodOptions, listOfInitOptions

---

**Note:** Make sure each parameter exists with your architecture and these lists : listOfEditCostOptions, listOfMethodOptions, listOfInitOptions. The structure of graphs must be similar as GXL.

---

PythonGedLib.**computeGedOnTwoGraphs** ()

Computes the edit distance between two NX graphs.

**Parameters**

- **g1** (*networksx.graph*) – The first graph to add and compute
- **g2** (*networksx.graph*) – The second graph to add and compute
- **editCost** (*string*) – The name of the edit cost function
- **method** (*string*) – The name of the computation method

- **options** (*string*) – The options of the method (like bash options), an empty string by default
- **initOption** (*string*) – The name of the init option, “EA-GER\_WITHOUT\_SHUFFLED\_COPIES” by default

**Returns** The edit distance between the two graphs and the nodeMap between them.

**Return type** double, list[tuple(size\_t, size\_t)]

**See also:**

listOfEditCostOptions, listOfMethodOptions, listOfInitOptions

---

**Note:** Make sure each parameter exists with your architecture and these lists : listOfEditCostOptions, listOfMethodOptions, listOfInitOptions. The structure of graphs must be similar as GXL.

---

PythonGedLib.**encodeYourMap**()

Encodes a string dictionary to utf-8 for C++ functions

**Parameters** **map** (*dict{string : string}*) – The map to encode

**Returns** The encoded map

**Return type** dict{‘b’string : ‘b’string}

---

**Note:** This function is used for type connection.

---



## HOW TO ADD YOUR OWN EDITCOST CLASS

When you choose your cost function, you can decide some parameters to personalize the function. But if you have some graphs which its type doesn't correspond to the choices, you can create your edit cost function.

For this, you have to write it in C++.

### 2.1 C++ side

Your class must inherit to EditCost class, which is an abstract class. You can find it here : `include/gedlib-master/src/edit_costs`

You can inspire you to the others to understand how to use it. You have to override these functions :

- `virtual double node_ins_cost_fun(const UserNodeLabel & node_label) const final;`
- `virtual double node_del_cost_fun(const UserNodeLabel & node_label) const final;`
- `virtual double node_rel_cost_fun(const UserNodeLabel & node_label_1, const UserNodeLabel & node_label_2) const final;`
- `virtual double edge_ins_cost_fun(const UserEdgeLabel & edge_label) const final;`
- `virtual double edge_del_cost_fun(const UserEdgeLabel & edge_label) const final;`
- `virtual double edge_rel_cost_fun(const UserEdgeLabel & edge_label_1, const UserEdgeLabel & edge_label_2) const final;`

You can add some attributes for parameters use or more functions, but these are unavoidable.

When your class is ready, please go to the C++ Bind here : `src/GedLibBind.cpp` . The function is :

```
void setPersonalEditCost(std::vector<double> editCostConstants){env.set_edit_costs(Your EditCost  
Class(editCostConstants));}
```

You have just to initialize your class. Parameters aren't mandatory, empty by default. If your class doesn't have one, you can skip this. After that, you have to recompile the project.

### 2.2 Python side

For this, use `setup.py` with this command in a linux shell:

```
python3 setup.py build_ext --inplace
```

You can also make it in Python 2.

Now you can use your edit cost function with the Python function `PySetPersonalEditCost(editCostConstant)`.

If you want more informations on C++, you can check the documentation of the original library here : <https://github.com/dbblumenthal/gedlib>

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`





**p**

PythonGedLib, 1



## A

addNxGraph() (in module PythonGedLib), 13  
 addRandomGraph() (in module PythonGedLib), 13  
 appel() (in module PythonGedLib), 13

## C

computeEditDistanceOnGXIIGraphs() (in module PythonGedLib), 13  
 computeEditDistanceOnNxGraphs() (in module PythonGedLib), 14  
 computeGedOnTwoGraphs() (in module PythonGedLib), 14

## E

EditCostError, 1  
 encodeYourMap() (in module PythonGedLib), 15  
 Error, 1

## I

InitError, 1

## M

MethodError, 1

## P

PyAddEdge() (in module PythonGedLib), 1  
 PyAddGraph() (in module PythonGedLib), 2  
 PyAddNode() (in module PythonGedLib), 2  
 PyAddSymmetricalEdge() (in module PythonGedLib), 2  
 PyClearGraph() (in module PythonGedLib), 2  
 PyGetAllGraphIds() (in module PythonGedLib), 3  
 PyGetAllMap() (in module PythonGedLib), 3  
 PyGetAssignmentMatrix() (in module PythonGedLib), 3  
 PyGetBackwardMap() (in module PythonGedLib), 4  
 PyGetDummyNode() (in module PythonGedLib), 4  
 PyGetEditCostOptions() (in module PythonGedLib), 4  
 PyGetForwardMap() (in module PythonGedLib), 4  
 PyGetGraphAdjacenceMatrix() (in module PythonGedLib), 5  
 PyGetGraphClass() (in module PythonGedLib), 5  
 PyGetGraphEdges() (in module PythonGedLib), 5  
 PyGetGraphIds() (in module PythonGedLib), 5

PyGetGraphInternalId() (in module PythonGedLib), 6  
 PyGetGraphName() (in module PythonGedLib), 6  
 PyGetGraphNodeLabels() (in module PythonGedLib), 6  
 PyGetGraphNumEdges() (in module PythonGedLib), 6  
 PyGetGraphNumNodes() (in module PythonGedLib), 6  
 PyGetInitime() (in module PythonGedLib), 7  
 PyGetInitOptions() (in module PythonGedLib), 7  
 PyGetLowerBound() (in module PythonGedLib), 7  
 PyGetMethodOptions() (in module PythonGedLib), 7  
 PyGetNodeImage() (in module PythonGedLib), 8  
 PyGetNodeMap() (in module PythonGedLib), 8  
 PyGetNodePreImage() (in module PythonGedLib), 9  
 PyGetOriginalNodeIds() (in module PythonGedLib), 9  
 PyGetRuntime() (in module PythonGedLib), 9  
 PyGetUpperBound() (in module PythonGedLib), 10  
 PyHungarianLSAP() (in module PythonGedLib), 10  
 PyHungarianLSAPE() (in module PythonGedLib), 10  
 PyInitEnv() (in module PythonGedLib), 10  
 PyInitMethod() (in module PythonGedLib), 11  
 PyIsInitialized() (in module PythonGedLib), 11  
 PyLoadGXLGraph() (in module PythonGedLib), 11  
 PyQuasimetricCost() (in module PythonGedLib), 11  
 PyRestartEnv() (in module PythonGedLib), 11  
 PyRunMethod() (in module PythonGedLib), 12  
 PySetEditCost() (in module PythonGedLib), 12  
 PySetMethod() (in module PythonGedLib), 12  
 PySetPersonalEditCost() (in module PythonGedLib), 12  
 PythonGedLib (module), 1