

---

# **absorbing-centrality**

***Release***

February 02, 2016



<b>1</b>	<b>Overview</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
<b>3</b>	<b>Usage</b>	<b>7</b>
<b>4</b>	<b>Reference</b>	<b>9</b>
<b>5</b>	<b>Contributing</b>	<b>15</b>
<b>6</b>	<b>Authors</b>	<b>17</b>
<b>7</b>	<b>Changelog</b>	<b>19</b>
<b>8</b>	<b>Indices and tables</b>	<b>21</b>



Contents:



---

## Overview

---

### 1.1 Absorbing Random-Walk Centrality

This is an implementation of the *absorbing random-walk centrality* measure for nodes in graphs. For the definition of the measure, as well as a study of the related optimization problem and algorithmic techniques, please see the pre-print publication on [arXiv](#). A short version of this paper will appear in the [ICDM 2015](#).

To cite this work, please use

```
Mavroforakis, Charalampos, Michael Mathioudakis, and Aristides Gionis.  
"Absorbing random-walk centrality: Theory and algorithms"  
Data Mining (ICDM), 2015 IEEE International Conference on. IEEE, 2015.
```

#### 1.1.1 Installation

You can install the *absorbing centrality* package by executing the following command in a terminal.

```
pip install absorbing centrality
```

#### 1.1.2 Documentation

For instructions on how to use the package, consult [its documentation](#).

#### 1.1.3 Example

You can find an example of how to use this package in [this IPython notebook](#).

#### 1.1.4 Development

To run all the tests for the code, you will need [tox](#) – check its webpage for instructions on how to install it.

Once [tox](#) is installed, use your terminal to enter the directory with the local copy of the code (here it's named '*absorbing-centrality*') and simply type the following command.

```
absorbing-centrality $ tox
```

If everything goes well, you'll receive a congratulatory message.

Note that the code is distributed under the Open Source Initiative (ISC) license. For the exact terms of distribution, see the LICENSE.

```
Copyright (c) 2015, absorbing-centrality contributors,  
Charalampos Mavroforakis <cmav@bu.edu>,  
Michael Mathioudakis <michael.mathioudakis@aalto.fi>,  
Aristides Gionis <aristides.gionis@aalto.fi>
```

---

### Installation

---

At the command line:

```
pip install absorbing_centrality
```



---

### Usage

---

To use Absorbing Random-Walk Centrality in a project:

```
import absorbing_centrality
```



## Reference

## 4.1 Computing the centrality of a set

---

<code>absorbing_centrality(G, team[, query, P, ...])</code>	Compute the absorbing centrality of a team.
<code>absorbing_centrality_inversion(G, team[, ...])</code>	Compute the absorbing centrality of a team using a fast inversion with S

---

### 4.1.1 `absorbing_centrality.absorbing_centrality`

**absorbing\_centrality**(*G*, *team*, *query*=None, *P*=None, *epsilon*=1e-05, *max\_iterations*=None, *with\_restarts*=False, *alpha*=0.85)

Compute the absorbing centrality of a team. The algorithm works by iteratively computing the powers of the non-absorbing submatrix of the transition matrix *P*.

#### Parameters

- **G** (*NetworkX graph*) – The graph on which to compute the centrality.
- **team** (*list*) – The team of nodes, whose centrality to compute.
- **query** (*list, optional*) – The set of query nodes to use for the random walks. If None (default) or empty, the query set is equal to the set of all nodes in the graph.
- **P** (*matrix, optional*) – The precomputed transition matrix of the graph (default is None).
- **epsilon** (*float, optional*) – The iterative algorithm stops when the error between the centrality computed by two successive iterations falls below epsilon (default is 1e-5).
- **max\_iterations** (*int, optional*) – The upper limit to the number of iterations of the algorithm (default is None).
- **with\_restarts** (*bool, optional*) – If True, restarts the random surfer to the the query set (default is False).
- **alpha** (*float, optional*) – The probability of the random surfer to continue (default is 0.85).

**Returns** *score* – The absorbing centrality score.

**Return type** *float*

---

**Note:** Both *team* and *query* should use the original node names.

---

### 4.1.2 absorbing\_centrality.absorbing\_centrality\_inversion

**absorbing\_centrality\_inversion** (*G*, *team*, *query=None*, *with\_restarts=False*, *alpha=0.85*)

Compute the absorbing centrality of a team using a fast inversion with SuperLU solver.

**Parameters**

- **G** (*NetworkX graph*) – The graph on which to compute the centrality.
- **team** (*list*) – The team of nodes, whose centrality to compute.
- **query** (*list, optional*) – The set of query nodes to use for the random walks. If *None* (default) or empty, the query set is equal to the set of all nodes in the graph.
- **with\_restarts** (*bool, optional*) – If *True*, restarts the random surfer to the the query set (default is *False*).
- **alpha** (*float, optional*) – The probability of the random surfer to continue (default is 0.85).

**Returns** *score* – The absorbing centrality score.

**Return type** *float*

---

**Note:** Both *team* and *query* should use the original node names.

---

## 4.2 Team-selection algorithms

---

*greedy\_team*(*G*, *k*[, *query*, *candidates*, ...]) Selects a team of nodes according to the greedy algorithm.

---

### 4.2.1 absorbing\_centrality.greedy\_team

**greedy\_team** (*G*, *k*, *query=None*, *candidates=None*, *fast\_select=False*, *return\_times=False*, *with\_restarts=False*, *alpha=0.85*)

Selects a team of nodes according to the greedy algorithm.

**Parameters**

- **G** (*Networkx graph*) – The graph from which the team will be selected.
- **k** (*int*) – The size of the team.
- **query** (*list, optional*) – If provided, the distance is measured with respect to the nodes in *query*.
- **candidates** (*list, optional*) – If provided, the team is picked only among the nodes in *candidates*.
- **fast\_select** (*bool, optional*) – If *True*, the greedy algorithm will only consider candidates in a smart way, by examining their gain at each round (default is *False*).
- **with\_restarts** (*bool, optional*) – If *True*, the greedy algorithm is based on the transition matrix w/ restarts to the supernode (default is *False*).
- **alpha** (*float, optional*) – If the transition matrix has restarts, *alpha* is the probability for the random surfer to continue (default is 0.85).

**Returns**

- **scores** (*list*) – The scores of all the greedy teams of size up to  $k$ .
- **teams** (*list*) – The list of greedy times of size up to  $k$ .
- **times** (*list*) – The time to compute each team. Returned only if *return\_times* is True.

## 4.3 Matrix-related functions

<code>compute_fundamental_matrix(P[, fast, ...])</code>	Computes the fundamental matrix for an absorbing random walk.
<code>compute_transition_matrix(G)</code>	Builds the random transition matrix P.
<code>compute_personalized_transition_matrix(G[, ...])</code>	Returns the transition matrix of the random walk with restarts.
<code>update_fundamental_matrix(P, F, next, previous)</code>	Applies Woodbury's formula to update the fundamental matrix in

### 4.3.1 absorbing\_centralty.compute\_fundamental\_matrix

**compute\_fundamental\_matrix** ( $P$ , *fast=True*, *drop\_tol=1e-05*, *fill\_factor=1000*)

Computes the fundamental matrix for an absorbing random walk.

#### Parameters

- **P** (*scipy.sparse matrix*) – The transition probability matrix of the absorbing random walk. To construct this matrix, you start from the original transition matrix and delete the rows that correspond to the absorbing nodes.
- **fast** (*bool, optional*) –
- **True (default), use the iterative SuperLU solver from (If)** –
- **scipy.sparse.linalg.** –
- **drop\_tol** (*float, optional*) – If *fast* is True, the *drop\_tol* parameter of the SuperLU solver is set to this value (default is 1e-5).
- **fill\_factor** (*int, optional*) – If *fast* is True, the *fill\_factor* parameter of the SuperLU solver is set to this value (default is 1000).

**Returns F** – The fundamental matrix of the random walk. Element (i,j) holds the expected number of times the random walk will be in state j before absorption, when it starts from state i. For more information, check <sup>1</sup>.

**Return type** `scipy.sparse matrix`

#### References

### 4.3.2 absorbing\_centralty.compute\_transition\_matrix

**compute\_transition\_matrix** ( $G$ )

Builds the random transition matrix P. The probability of going from node  $i$  to node  $j$  is equal to:

$$P_{i,j} = \frac{1}{\text{degree}(i)}$$

**Parameters G** (*NetworkX graph*) –

<sup>1</sup> Doyle, Peter G., and J. Laurie Snell. Random walks and electric networks. Carus mathematical monographs 22 (2000). <https://math.dartmouth.edu/~doyle/docs/walks/walks.pdf>

**Returns** **P** – The random transition probability matrix.

**Return type** `scipy.sparse matrix`

### 4.3.3 `absorbing_centrality.compute_personalized_transition_matrix`

**`compute_personalized_transition_matrix`** (*G*, *alpha*=0.85, *restart\_set*=['\_super\_'])

Returns the transition matrix of the random walk with restarts.

#### Parameters

- **G** (*graph*) –
- **alpha** (*float*, *optional*) – The probability of the random surfer to continue their walk (default is 0.85).
- **restart\_set** (*list*, *optional*) – The set of nodes to restart from. If not supplied, the restarts lead to the supernode (default is [SUPER\_NODE]).

**Returns** **P** – The probability matrix for the random walk with restarts.

**Return type** `scipy.sparse.matrix`

### 4.3.4 `absorbing_centrality.update_fundamental_matrix`

**`update_fundamental_matrix`** (*P*, *F*, *next*, *previous*, *previous\_index*=0, *node\_order*=None)

Applies Woodbury's formula to update the fundamental matrix in order to avoid doing an inversion.

#### Parameters

- **P** (*matrix*) – The transition matrix of the graph, where *previous* is non absorbing.
- **F** (*matrix*) – The fundamental matrix of the graph after setting the node *previous* as an absorbing node.
- **next** (*int*) – The node that will be set as absorbing next. The result of this call will result in a fundamental matrix where *next* is an absorbing node.
- **previous** (*int*) – The node that was set as absorbing when computing F.
- **previous\_index** (*int*, *optional*) – The row/col index of node *previous* in P (default is 0).
- **node\_order** (*list*, *optional*) – The nodes that corresponds to the rows/cols of *P*, in order. If not supplied, the order is considered to be [0, .. , n\_P - 1], where n\_P is the the number of rows/cols in *P*.

#### Returns

- **P\_updated** (*matrix*) – The new transition matrix, where *previous* is absorbing.
- **F\_updated** (*matrix*) – The fundamental matrix after adding *previous* and *next* in the set of absorbing nodes.
- **node\_order\_updated** (*list*) – The new order of the non absorbing nodes in the *F\_new*.
- **next\_index** (*int*) – The row/col index of the node *next*, that we just set as absorbing, in *P\_new*.

## 4.4 Preprocessing the graph

<code>canonical_relabel_nodes(G)</code>	Relabels the nodes in the graph, such that the new names belong in the set [1,n].
<code>is_canonical(G)</code>	Tests if the graph has been canonicalized.
<code>add_supernode(G[, query])</code>	Adds a supernode to the graph and connects it with directed edges to the query nodes.
<code>has_supernode(G)</code>	Checks if there exist a supernode in the graph.

### 4.4.1 absorbing\_centralty.canonical\_relabel\_nodes

**canonical\_relabel\_nodes** (*G*)

Relabels the nodes in the graph, such that the new names belong in the set [1,n]. The labeling information is stored in the dictionaries *G.graph['canonical\_map']* and *G.graph['label\_map']*. These provide a way to map original to canonical node names and vice-versa, respectively.

**Parameters** *G* (*NetworkX graph*) –

**Returns**

**G\_prime** – The relabeled graph. It includes two attributes:

1. *G\_prime.graph['canonical\_map']* [dict] Holds the mapping between the original names of the nodes and the new, canonical, names (original -> new).
2. *G\_prime.graph['label\_map']* [dict] Holds the mapping between the new, canonical, names and the original names of the nodes (new -> original).

**Return type** NetworkX graph

Note: The relabeling of a particular node might not be consistent across two consecutive runs. Also, the relabeling happens on a copy, so the original graph will be untouched.

### 4.4.2 absorbing\_centralty.is\_canonical

**is\_canonical** (*G*)

Tests if the graph has been canonicalized.

**Parameters** *G* (*NetworkX graph*) –

**Returns**

- *bool*
- Returns True, if the graph has been canonicalized.

### 4.4.3 absorbing\_centralty.add\_supernode

**add\_supernode** (*G*, *query=None*)

Adds a supernode to the graph and connects it with directed edges to the query nodes.

**Parameters**

- *G* (*NetworkX graph*) – The graph in which we want to add a supernode.
- **query** (*list*, *default is None*) – The list of nodes that the supernode will be connected to. If *query* is None, the supernode will be connected to all the nodes in *G*. These new edges will be directed.

**Returns** A directed graph with the supernode, and the new edges, added. The attributes of the graph, i.e. 'label\_map' and 'canonical\_map', are also updated (or created if the input graph was not canonicalized) to reflect the new node.

**Return type** NetworkX graph

#### 4.4.4 `absorbing_centrality.has_supernode`

**has\_supernode** (*G*)

Checks if there exist a supernode in the graph.

**Parameters** *G* (*NetworkX graph*) –

**Returns** `has_supernode`

**Return type** `bool`

### 4.5 Exceptions

---

`CanonicalizationError`(message) Exception related to the graph canonicalization procedure.

---

#### 4.5.1 `absorbing_centrality.CanonicalizationError`

**exception `CanonicalizationError`** (*message*)

Exception related to the graph canonicalization procedure.

---

## Contributing

---

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

### 5.1 Bug reports

When [reporting a bug](#) please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

### 5.2 Documentation improvements

Absorbing Random-Walk Centrality could always use more documentation, whether as part of the official Absorbing Random-Walk Centrality docs, in docstrings, or even on the web in blog posts, articles, and such.

### 5.3 Feature requests and feedback

The best way to send feedback is to file an issue at <https://github.com/harrymvr/absorbing-centrality/issues>.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

### 5.4 Development

To set up *absorbing-centrality* for local development:

1. [Fork absorbing-centrality on GitHub](#).
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/absorbing-centrality.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with `tox` one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

### 5.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)<sup>1</sup>.
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

### 5.4.2 Tips

To run a subset of tests:

```
tox -e envname -- py.test -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

---

<sup>1</sup> If you don't have all the necessary python versions available locally you can rely on Travis - it will [run the tests](#) for each change you add in the pull request.

It will be slower though ...

---

### Authors

---

- Charalampos Mavroforakis - <http://cs-people.bu.edu/cmav>
- Michael Mathioudakis - <http://michalis.co>
- Aristides Gionis - <http://users.ics.aalto.fi/gionis/>



---

## Changelog

---

### 7.1 0.1.0 (2015-08-31)

- Working version of the package.



---

## Indices and tables

---

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



## A

`absorbing_centrality()` (in module `absorbing_centrality`), [9](#)  
`absorbing_centrality_inversion()` (in module `absorbing_centrality`), [10](#)  
`add_supernode()` (in module `absorbing_centrality`), [13](#)

## C

`canonical_relabel_nodes()` (in module `absorbing_centrality`), [13](#)  
`CanonicalizationError`, [14](#)  
`compute_fundamental_matrix()` (in module `absorbing_centrality`), [11](#)  
`compute_personalized_transition_matrix()` (in module `absorbing_centrality`), [12](#)  
`compute_transition_matrix()` (in module `absorbing_centrality`), [11](#)

## G

`greedy_team()` (in module `absorbing_centrality`), [10](#)

## H

`has_supernode()` (in module `absorbing_centrality`), [14](#)

## I

`is_canonical()` (in module `absorbing_centrality`), [13](#)

## U

`update_fundamental_matrix()` (in module `absorbing_centrality`), [12](#)