



Graphab 2.2

Reference Manual

Command Line Interface

Céline Clauzel, Jean-Christophe Foltête, Xavier Girardet, Gilles Vuidel

2017-09-11

Contents

1	Prerequisite	3
1.1	About	3
1.1.1	Authors	3
1.1.2	Terms of use	3
1.2	System requirements	3
1.3	Installing the software and launching a project	3
1.4	Launch Graphab in CLI mode	4
1.5	Syntax	4
1.5.1	Definition	4
1.5.2	Character separator	4
1.5.3	Optional parameter	4
1.5.4	Range and value list	4
1.5.5	Command execution	5
2	Command reference	6
2.1	General command	6
2.1.1	-help : display help	6
2.1.2	-metrics : list available metrics	7
2.1.3	-create : create project	7
2.1.4	-project : load project	8
2.1.5	-show : list project elements	8
2.1.6	-capa : set patch capacity	8
2.1.7	-metapatch : create meta-patch project	9
2.2	Graph management	10
2.2.1	-linkset : create cost linkset	10
2.2.2	-uselinkset : select linkset	11
2.2.3	-graph : create graph	11
2.2.4	-usegraph : select graph	11
2.2.5	-corridor : calculate corridors	12
2.2.6	-cluster : graph clustering	12
2.3	Calculate metric	13
2.3.1	-gmetric : calculate global metric	13
2.3.2	-cmetric : calculate component metric	14
2.3.3	-lmetric : calculate local metric	15
2.3.4	-interp : interpolate a metric	15
2.4	SDM	16
2.4.1	-pointset : import pointset	16
2.4.2	-usepointset : select pointset	17
2.4.3	-model : calculate SDM	17
2.5	Adding/Removal	17
2.5.1	-delta: remove one item	17
2.5.2	-gremove: remove several items	18
2.5.3	-gtest: removal and iterative item addition	19
2.5.4	-addpatch : iterative patch addition	21
2.5.5	-remelem: iterative item removal	22
2.6	Land use changes	23

2.6.1	-landmod : land use changes	23
2.7	Options	24
2.7.1	-nosave	24
2.7.2	-proc	24
2.7.3	-mpi	24
3	Command examples	25
3.1	Display project	25
3.2	Batch metric parameter	25
3.3	Batch thresholded graph and global metric	26
3.4	Complete SDM sequence	26
4	Performance tuning	27
4.1	Parallelism to speed up execution	27
4.1.1	One computer : threads	27
4.1.2	Computer cluster : mpi	27
4.2	Memory management	27

Chapter 1

Prerequisite

Graphab is a software application for modeling ecological networks using landscape graphs.

Graphab can be used in command line interface (CLI) from version 1.2. It is useful for executing graphab on a distant computer without a graphical interface, or batching some processes that are not available in the graphical user interface (GUI).

Warning, projects are not compatible between the versions 1.x and 2.x of Graphab !

1.1 About

1.1.1 Authors

Graphab has been developed by Gilles Vuidel and Jean-Christophe Foltête at [ThéMA](#) laboratory ([University of Franche-Comté – CNRS](#)). Funding has been provided by the French Ministry of Ecology, Energy, Sustainable Development and the Sea ([ITTECOP](#) Program). The Graphab logo was designed by [Gachwell](#).

1.1.2 Terms of use

Graphab is distributed in open source, under the GPL license. Users must cite the following reference [[Foltête et al.\(2012a\)](#)] in their publications:

Foltête J.C., Clauzel C., Vuidel G., 2012. A software tool dedicated to the modelling of landscape networks, *Environmental Modelling & Software*, 38: 316-327.

1.2 System requirements

Graphab runs on any computer supporting Java 7 or later (PC under Linux, Windows, Mac, etc.). However, when dealing with very large datasets, the amount of RAM memory in the computer will limit the maximum number of nodes and links that can be processed in a single run with Graphab. In addition, for some complex metrics, processing power (CPU) will determine the speed of computing. For details, see section ?? below and the journal article [[Foltête et al.\(2012a\)](#)].

1.3 Installing the software and launching a project

Graphab can be downloaded from <http://thema.univ-fcomte.fr/productions/graphab>.

- Download and install Java 7 or later - java.com. If you have a 64-bit operating system, it is best to install the 64-bit version of Java.

- Download graphab-2.2.jar
- Launch graphab-2.2.jar

1.4 Launch Graphab in CLI mode

First you have to open a terminal window. Then, go to the directory of the Graphab program with *cd* command. Finally, type the following command to display the Graphab help screen :

```
java -jar graphab-2.2.jar --help
```

Result:

```
Usage :
java -jar graphab.jar --project prjfile.xml
...
...
```

You're ready to use Graphab in CLI mode.

1.5 Syntax

1.5.1 Definition

Commands always start with a double dash: `--project`, `--linkset`, ...

A global option starts with only one dash: `-proc`, `-nosave`, ...

A parameter does not have a dash: `name`, `complete`, `maxcost`, ...

1.5.2 Character separator

Blank spaces are used to separate commands and parameters. You cannot have a name containing blank spaces.

Avoid blank spaces in the project's name and the project's elements.

1.5.3 Optional parameter

Parameters enclosed in brackets are optional. Therefore, parameters not in brackets are mandatory.

1.5.4 Range and value list

Defining a range of values for a given parameter (rather than a single value) executes the command several times for each value defined by the range. Ranges are defined by a minimum, increment and a maximum value, with inclusive bounds :

```
min:inc:max
```

A range from 0 to 10 by step of 2 will create 6 values : 0,2,4,6,8,10 :

```
0:2:10
```

The minimum value is always included, but the maximum value is included only if the incremented value falls exactly on the maximum, otherwise the last value will be the maximum incremented value smaller than the maximum value. A range from 0 to 9 by step of 2 will create 5 values : 0,2,4,6,8 :

```
0:2:9
```

Decimal values can be used, with a period for the decimal separator :

1.5:0.5:4

To process a non-consecutive set of parameter values, comma separated values can be used in place of a range :

0,1,4,8,10

The value list cannot contain blank spaces.

In all cases, a single value can be used instead of a range if we don't want several executions. If a command contains several ranges for different parameters, all combinations of ranges will be computed.

1.5.5 Command execution

Graphab can be launched with several commands on one line, except for the `--help` and `--metrics` commands. The `--create` and `--project` commands can be used only once and must be the first command. After one of these commands, all other commands will be executed sequentially with the same order as the command line.

```
java -jar graphab-2.2.jar --project prj.xml --graph --gmetric NC
```

Load a project, then execute the graph command and after that, execute the gmetric command.

Chapter 2

Command reference

2.1 General command

2.1.1 `--help` : display help

Command :

```
java -jar graphab-2.2.jar --help
```

Result :

Usage :

```
java -jar graphab.jar --metrics
```

```
java -jar graphab.jar [-proc n] --create prjname landrasterfile habitat=code1,...,coden ...
```

```
java -jar graphab.jar [-mpi | -proc n] [-nosave] --project prjfile.xml command1 [command2 ...]
```

Commands list :

```
--show
```

```
--linkset distance=euclid|cost [name=linkname] [complete=[dmax]] [slope=coef] [remcrosspath] ...
```

```
--uselinkset linkset1,...,linksetn
```

```
--corridor maxcost=[{}valcost[]]
```

```
--graph [name=graphname] [nointra] [threshold=[{}min:inc:max[]]]
```

```
--usegraph graph1,...,graphn
```

```
--cluster d=val p=val [beta=val] [nb=val]
```

```
--pointset pointset.shp
```

```
--usepointset pointset1,...,pointsetn
```

```
--capa [maxcost=[{}valcost[]] codes=code1,code2,...,coden [weight]]
```

```
--gmetric global_metric_name [maxcost=valcost] [param1=[{}min:inc:max[]] [param2=[{}min:inc:max[]] ...]
```

```
--cmetric comp_metric_name [maxcost=valcost] [param1=[{}min:inc:max[]] [param2=[{}min:inc:max[]] ...]
```

```
--lmetric local_metric_name [maxcost=valcost] [param1=[{}min:inc:max[]] [param2=[{}min:inc:max[]] ...]
```

```
--interp name resolution var=patch_var_name d=val p=val [multi=dist_max [sum]]
```

```
--model variable distW=min:inc:max
```

```
--delta global_metric_name [maxcost=valcost] [param1=val ...] obj=patch|link [sel=id1,id2, ...]
```

```
--addpatch npatch global_metric_name [param1=val ...] [gridres=min:inc:max [capa=capa_file] ...]
```

```
--remelem nstep global_metric_name [maxcost=valcost] [param1=val ...] obj=patch|link [sel=id1,id2, ...]
```

```
--gtest nstep global_metric_name [maxcost=valcost] [param1=val ...] obj=patch|link [sel=id1,id2, ...]
```

```
--gremove global_metric_name [maxcost=valcost] [param1=val ...] [patch=id1,id2,...,idn|fpatch= ...]
```

```
--metapatch [mincapa=value]
```

```
--landmod zone=filezones.shp id=fieldname code=fieldname [sel=id1,id2,...,idn ] [novoronoi]
```

min:inc:max -> val1,val2,val3...

2.1.2 `--metrics` : list available metrics

This command lists all available metrics with their short name, description, and applicable parameters if needed.

Command :

```
java -jar graphab-2.2.jar --metrics
```

Result :

```
===== Global metrics =====
S#F - Sum Flux (F)
      params : [d, p, beta]
PC - Probability of Connectivity (PC)
      params : [d, p, beta]
IIC - Integral index of connectivity (IIC)
CCP - Class coincidence probability (CCP)
MSC - Mean size of the components (MSC)
SLC - Size of the largest component (SLC)
ECS - Expected Cluster Size (ECS)
GD - Graph diameter (GD)
H - Harary index (H)
NC - Number of components (NC)
dPC - Delta PC decomposed (dPC)
      params : [d, p, beta]
W - Wilks index (W)
params : [attrs, npatch, warea]

===== Local metrics =====
F : Flux (F)
      params : [d, p, beta]
BC : Betweenness centrality (BC)
      params : [d, p, beta]
FPC : Flow PC (FPC)
      params : [d, p, beta]
Dg : Node degree (Dg)
CC : Clustering Coefficient (CC)
CCe : Closeness centrality (CCe)
CCor : Connectivity Correlation (CCor)
Ec : Eccentricity (Ec)
CF : Current flow (CF)
      params : [beta]
```

2.1.3 `--create` : create project

```
java -jar graphab-2.2.jar --create prjname landrasterfile habitat=code1,...,coden [nodata=val]
                               [minarea=val] [con8] [simp] [dir=path]
```

Required parameters

- `prjname`: name of the new project
- `landrasterfile`: filename containing land cover in tiff, ascii grid or rst format
- `habitat=code1,...,coden`: land codes for patch habitat.

Optional parameters

- **nodata=val**: code for nodata
- **minarea=val**: minimal patch size in hectare
- **con8**: neighborhood of 8 pixels for patch definition, by default the neighborhood is limited to 4 pixels
- **simp**: simplify patch geometry to speed up the planar topology creation (voronoï)
- **dir=path**: path for saving the project, by default the project is saved in the current directory.

Description

This commands creates a new project and loads it. The **--create** command can be used only once and must be the first command. The following commands will be executed on the new project.

2.1.4 **--project** : load project

This command sets the path to the project xml file.

```
java -jar graphab-2.2.jar --project path2myproject/myproject.xml
```

This command loads the project myproject contained in the path2myproject folder.

The **--project** command can be used only once and must be the first command. All of the following commands below require a loaded project.

2.1.5 **--show** : list project elements

Lists linksets, graphs and pointsets contained in the selected project. This command is useful to retrieve exact name of an element to be used in the command line.

Command :

```
java -jar graphab-2.2.jar --project path2myproject/myproject.xml --show
```

Result :

```
===== Link sets =====
Complete_Euclid
Complete_cost
Planar_Euclid
Planar_cost

===== Graphs =====
2000m-3500cost
2000m-3500cost_comp
2000m_euclid
2000m_euclid_comp

===== Point sets =====
Presence_absence
```

Pay attention to the element's name. The CLI is case sensitive and does not manage blank spaces in element names.

2.1.6 **--capa** : set patch capacity

```
--capa [maxcost=valcost codes=code1,code2,...,coden [weight]]
```

Optional parameters

- `maxcost=[{}valcost[]]`: maximum cost distance for the neighborhood
- `codes=code1,code2,...,coden`: land cover codes used to calculate the neighborhood area
- `weight`: introduce a weighting depending on the distance to the patch with a negative exponential function.

Description

The `--capa` command calculates the capacity of the patches and saves the project unless the `-nosave` option is used. Without parameter, the capacity is defined as the area of the patch in m^2 . With parameters, the capacity is calculated as the area of land cover elements around the patch, up to the distance *valcost*. The distances are calculated from the cost defined in the selected link set. If the link set is euclidean, costs are set to 1 for all land cover codes.

This command supports only one selected link set (cf. `-uselinkset : select linkset`).

Example

```
--capa maxcost=100 codes=1,3
```

Set the patch capacity to the area (in m^2) of land cover codes 1 and 3 in the neighborhood of the patch up to a distance of 100 (in cost unit).

2.1.7 `--metapatch : create meta-patch project`

```
--metapatch [mincapa=value]
```

Optional parameter

- `mincapa=value`: minimum capacity of a meta-patch to be retained in the new project

Description

The `--metapatch` command creates a new project with meta-patch based on the selected graph. Each component of the graph becomes a meta-patch. This meta-patch is composed of all patches contained in the component, ie. all patches connected together. The capacity of a meta-patch is set to the sum of the capacity of the patches composing the meta-patch. The new project is saved in a subdirectory of the current project.

This new project becomes the current project for the next commands.

This command supports only one selected graph (cf. `-usegraph : select graph`).

Example

```
--usegraph 2000m_euclid --metapatch mincapa=1000
```

This command creates a meta-patch project based on the graph *2000m_euclid* and removes all meta-patches with a capacity lower than 1000.

References

[[Clauzel et al.\(2015b\)](#)], [[Foltête et al.\(2016\)](#)]

2.2 Graph management

2.2.1 `--linkset` : create cost linkset

```
--linkset distance=euclid|cost [name=linkname] [complete[=dmax]] [slope=coef] [remcrosspath]
[[code1,...,coden=cost1 ...] codei,...,codej=min:inc:max | extcost=rasterfile]
```

Optional parameters

- `name=linkname`: link set name to create
- `complete[=dmax]`: set to complete topology in place of planar topology. The *dmax* value can be added to limit path calculation up to the given distance *dmax*.

Parameters for the case `distance=cost`

- `slope=coef`: weight costs with the slope. For using this option, the project must contain a DEM.
- `remcrosspath`: remove links crossing patches
- [*code1*,...,*coden*=*cost1* ...] *codei*,...,*codej*=*min:inc:max*: set the cost for each land cover codes
- `extcost=rasterfile`: set the costs from an external raster in tiff, ascii grid or rst format.

Description

Creates a new linkset in the loaded project and saves the project unless the `--nosave` option is used.

If the `name` parameter is not set, the linkset's name is derived from the cost definition.

The `--linkset` command does not accept several ranges.

After `--linkset` command execution, ensuing linkset selection is set to created linksets.

Examples

This command will create a planar linkset *cost_1_2_3_4_5_6_7-1.0* with all costs equal to 1:

```
--linkset distance=cost 1,2,3,4,5,6,7=1
```

This command will create a planar linkset *cost_1_2_3-1.0* with cost equal to 1 for landscape values 1, 2 and 3, and cost equal to 2 for landscape values 4, 5, 6 and 7:

```
--linkset distance=cost 1,2,3=1 4,5,6,7=2
```

The default topology is planar; use the `complete` option to create a complete topology linkset:

```
--linkset distance=cost complete 1,2,3,4,5,6,7=1
```

With a complete topology, you can prescribe a threshold to avoid the creation of too many links (threshold = 100):

```
--linkset distance=cost complete=100 1,2,3,4,5,6,7=1
```

A range or value list can be used to create multiple linksets:

```
--linkset distance=cost 4,5,6,7=10 1,2,3=100:50:200
```

or

```
--linkset distance=cost 4,5,6,7=10 1,2,3=100,150,200
```

Result: the command above created 3 linksets *cost_1_2_3-100.0* *cost_1_2_3-150.0* *cost_1_2_3-200.0* where raster codes 1,2 and 3 were 100, 150 and 200 respectively.

2.2.2 `--uselinkset : select linkset`

`--uselinkset linkset1,...,linksetn`

Selects linksets to be used in following commands.

The linkset name is case sensitive and must not contain blank spaces.

By default, all linksets are selected.

2.2.3 `--graph : create graph`

`--graph [name=graphname] [nointra] [threshold=[{]min:inc:max[}]]`

Optional parameters

- **name=graphname** : name of the graph created. This parameter can be used only if this command creates only one graph.
- **nointra**: disable intra-patch distances for metric calculation
- **threshold=[{]min:inc:max[}]**: set the maximum distance for links included in the graph. Without this parameter, the graph contains all links of the selected link set. Surrounded by braces, the distance values are converted automatically in cost values.

Description

Creates a graph from selected linksets and saves the project unless `--nosave` option is used. Currently, this command does not permit the Minimum Spanning Tree option.

After `--graph` command execution, ensuing graph selection is set to the created graphs.

Examples

Without set parameters, the command will create one graph without a threshold for each selected linkset:

```
--graph
```

The graph name will be the concatenation of `comp_` and linkset name.

If a unique value threshold is specified, the command will create one graph with the given threshold for each selected linkset:

```
--graph threshold=100
```

The graph name will be the concatenation of `thresh_100.0_` and the linkset name.

If the threshold parameter is defined with a value list or a range, it will create a thresholded graph for each linkset and each threshold:

```
--graph threshold=1000:100:1500
```

or

```
--graph threshold=1000,1100,1200,1300,1400,1500
```

Result: this command created 6 graphs for each selected linkset.

2.2.4 `--usegraph : select graph`

`--usegraph graph1,...,graphn`

Selects graphs to be used in following commands.

The graph name is case sensitive and must not contain blank spaces.

By default, all graphs are selected.

2.2.5 `--corridor` : calculate corridors

```
--corridor maxcost=valcost
```

Required parameter

- `maxcost=[{valcost}]`: maximum cost paths forming a corridor. Surrounded by braces, the distance given in meter is converted automatically in cost unit.

Description

The `--corridor` command calculates the corridor associated to each link of a link set. The result is stored in a shapefile for each selected link set. For each link, the shapefile contains a polygon representing the set of paths having a distance less than or equal to *valcost*. This command does not operate on euclidean link sets, they are ignored.

Example

```
--corridor maxcost=500
```

2.2.6 `--cluster` : graph clustering

```
--cluster d=val p=val [beta=val] [nb=val]
```

Required parameters

- `d=val` : distance for setting α exponent.
- `p=val` : probability for setting α exponent. For setting α to 0 and ignore links impedance, sets `p` to 1.

Optional parameters

- `beta=val` : capacity exponent, by default set to 1. To ignore patch capacity, set `beta` to 0.
- `nb=val` : number of compartments, by default the number of compartments selected corresponds to the maximum modularity.

Description

Creates a graph based on the clustering which maximises the modularity [Newman(2006)] for each selected graph and saves the project unless the global option `-nosave` is used. The new graphs keep only intra-cluster links.

Modularity is calculated with a weight(w_{ij}) defined for each link of the graph:

$$w_{ij} = (a_i a_j)^\beta e^{-\alpha d_{ij}}$$

The two parameters β et α are used to define respectively the importance of the patch capacity ($a_i a_j$) and the importance of the distance (d_{ij}) for the weight of the link w_{ij} . If $\alpha = \beta = 0$, then the weights are identical: $w_{ij} = 1$.

After `--cluster` command, the selected graphs correspond to the newly created graphs.

Reference

[Foltête and Vuidel(2017)]

2.3 Calculate metric

2.3.1 `--gmetric` : calculate global metric

```
--gmetric global_metric_name [maxcost=valcost] [param1=min:inc:max [param2=min:inc:max ...]]
```

Required parameter

- `global_metric_name` : global metric short's name (PC, IIC, ...)

Optional parameters

- `maxcost=valcost` : limit pathfinding to *valcost* (ie. paths greater than *maxcost* are not calculated). It can reduce the time execution for path-based metric but results may be inaccurate.
- `param1=[{}min:inc:max[]]` : metric parameter(s), if needed. Surrounded by braces, the distance values are converted automatically in cost values.
- ...

Description

Calculates a given global metric on each selected graph. The metric's name is the short name as shown in `--metrics` command. If the metric requires parameters, they can be specified in any order. Results are stored in a text file in the project folder. The file name corresponds to the metric short name.

Examples

To calculate the NC metric on selected graphs:

```
--gmetric NC
```

Results are stored in the file NC.txt in the project folder:

Graph	NC
2000m-3500cost	25.0
2000m-3500cost_comp	24.0
2000m_euclid	9.0
2000m_euclid_comp	9.0

For metrics requiring parameters, you can test several sets of parameters in one command. The following command executes 6 PC metrics for each graph with the parameter `d` equal to 1000,1500 or 2000 and `beta` equal to 0 or 1 :

```
--gmetric PC d=1000:500:2000 p=0.05 beta=0,1
```

Results are stored in file the PC.txt :

Graph	d	p	beta	PC
2000m-3500cost	1000.0	0.05	0.0	2.108166945899072E-15
2000m-3500cost	1500.0	0.05	0.0	2.4839095790785042E-15
2000m-3500cost	2000.0	0.05	0.0	2.866220685546806E-15
2000m-3500cost	1000.0	0.05	1.0	1.317091007462398E-6

2000m-3500cost	1500.0	0.05	1.0	1.4758311225154786E-6
2000m-3500cost	2000.0	0.05	1.0	1.5884005111333579E-6
2000m-3500cost_comp	1000.0	0.05	0.0	2.1106833149811817E-15
2000m-3500cost_comp	1500.0	0.05	0.0	2.493027588606987E-15
2000m-3500cost_comp	2000.0	0.05	0.0	2.887027144684246E-15
2000m-3500cost_comp	1000.0	0.05	1.0	1.3171878007185563E-6
2000m-3500cost_comp	1500.0	0.05	1.0	1.476224502635306E-6
2000m-3500cost_comp	2000.0	0.05	1.0	1.5892024206564504E-6
2000m_euclid	1000.0	0.05	0.0	2.8238137481213476E-15
2000m_euclid	1500.0	0.05	0.0	3.516516320195261E-15
2000m_euclid	2000.0	0.05	0.0	4.285009196943927E-15
2000m_euclid	1000.0	0.05	1.0	1.7079340030649911E-6
2000m_euclid	1500.0	0.05	1.0	1.8176869551880345E-6
2000m_euclid	2000.0	0.05	1.0	1.8976284261240914E-6
2000m_euclid_comp	1000.0	0.05	0.0	2.867215798466552E-15
2000m_euclid_comp	1500.0	0.05	0.0	3.581685718161269E-15
2000m_euclid_comp	2000.0	0.05	0.0	4.374805768414666E-15
2000m_euclid_comp	1000.0	0.05	1.0	1.7172345329380555E-6
2000m_euclid_comp	1500.0	0.05	1.0	1.8277346595840518E-6
2000m_euclid_comp	2000.0	0.05	1.0	1.9080569002930505E-6

2.3.2 `--cmetric` : calculate component metric

```
--cmetric global_metric_name [maxcost=valcost] [param1=min:inc:max [param2=min:inc:max ...]]
```

Required parameter

- `global_metric_name` : global metric short's name (PC, IIC, ...)

Optional parameters

- `maxcost=valcost` : limit pathfinding to *valcost* (ie. paths greater than *maxcost* are not calculated). It can reduce the time execution for path-based metric but results may be inaccurate.
- `param1=[{}min:inc:max[]]` : metric parameter(s), if needed. Surrounded by braces, the distance values are converted automatically in cost values.
- ...

Description

Calculates a given global metric on each component on each selected graph. The result is saved in the project unless `--nosave` option is used. The `--nosave` option is useful only with the `--model` or `--interp` command.

Example

Whith metrics requiring parameters, you can test several sets of parameters in one command.

```
--cmetric PC d=1000:500:2000 p=0.05 beta=0,1
```

Six PC are calculated for each component of each selected graph :

- PC_d1000_p0.05.beta0
- PC_d1500_p0.05.beta0

- PC_d2000_p0.05_beta0
- PC_d1000_p0.05_beta1
- PC_d1500_p0.05_beta1
- PC_d2000_p0.05_beta1

2.3.3 `--lmetric` : calculate local metric

```
--lmetric local_metric_name [maxcost=valcost] [param1=min:inc:max [param2=min:inc:max ...]]
```

Required parameter

- `local_metric_name` : local metric short's name (F, CF, ...)

Optional parameters

- `maxcost=valcost` : limit pathfinding to *valcost* (ie. paths greater than *maxcost* are not calculated). It can reduce the time execution for path-based metric but results may be inaccurate.
- `param1=[{}min:inc:max{}]` : metric parameter(s), if needed. Surrounded by braces, the distance values are converted automatically in cost values.
- ...

Description

Calculates a given local metric on each selected graph. The result is saved in the project unless `-nosave` option is used. The `-nosave` option is useful only with the `--model` or `--interp` command.

Example

With metrics requiring parameters, you can test several sets of parameters in one command.

```
--lmetric F d=1000:500:2000 p=0.05 beta=0,1
```

Six F metrics are calculated for each selected graph :

- F_d1000_p0.05_beta0
- F_d1500_p0.05_beta0
- F_d2000_p0.05_beta0
- F_d1000_p0.05_beta1
- F_d1500_p0.05_beta1
- F_d2000_p0.05_beta1

2.3.4 `--interp` : interpolate a metric

```
--interp name resolution var=patch_var_name d=val p=val [multi=dist_max [sum]]
```


Required parameters

- **name** : name of the raster file storing the result of the interpolation
- **resolution** : resolution of the interpolation. In some cases, this parameter is ignored and the resolution is set to the land cover map resolution
- **var=patch_var_name** : name of the variable to interpolate. This name corresponds to a patch attribute ; usually the result of a local metric calculation.
- **d=val** : distance allowing, with the **p** parameter, to set the coefficient α
- **p=val** : probability allowing, with the **d** parameter, to set the coefficient α

Optional parameters

- **multi=dist_max** : interpolation from all patches within the *dist_max* distance, instead of the nearest patch only
- **sum** : aggregation by a sum rather than a weighted average. Used only with the **multi** parameter.

Description

The **--interp** command allows to interpolate on the whole area, a data available only on patches level, usually a local metric. The interpolation is based on a decreasing function of the distance. The interpolated value at a point p_j is defined as :

$$p_j = var_i * e^{-\alpha d_{ij}}$$

var_i is the variable value of the closest patch from the point p_j . The α coefficient is defined by the 2 parameters **d** et **p**, such as $p = e^{-\alpha d}$. The distance d_{ij} calculation between the point and the closest patch depends on the current link set (euclidean or cost).

If the **multi** parameter is enabled, the interpolation is not calculated from the closest patch only, but also from all the patches whose distance is less than or equal to *dist_max*. This several values are aggregated by an weighted mean, by default. If the **sum** parameter is added, the aggregation used is a sum. In the latter case, the interpolation formula of a point p_j is :

$$p_j = \sum_i var_i * e^{-\alpha d_{ij}}$$

If several link sets are selected, a interpolation is calculated for each link set.

Example

```
--interp test 10 var=F_d1000_p0.5_beta1_2000m_euclid d=1000 p=0.5 multi=1000 sum
```

2.4 SDM

2.4.1 --pointset : import pointset

```
--pointset pointset.shp
```

Creates a new pointset from the shapefile parameter for each selected linkset and saves the project unless **-nosave** option is used. The pointset name is a concatenation of the shapefile name and the linkset name.

After **--pointset** command execution, pointset selection is set to the created pointsets only.

2.4.2 `--usepointset` : select pointset

`--usepointset ps1,...,psn`

Selects pointsets to be used in the following `--model` command. By default, all pointsets are selected.

2.4.3 `--model` : calculate SDM

`--model variable distW=min:inc:maxcost`

Required parameters

- **variable**: name of a binary variable contained in the point set(s)
- **distW=min:inc:maxcost**: distance weighting between patch and point with probability 0.05

Description

Calculates SDM on each metric existing in all selected graphs for the binary pointset *variable*. The *variable* must exist in all selected pointsets. For each graph, the command search for a pointset which have the same linkset as the graph. If none exists, it will throw an error, if several exist, it will use any. The parameter **distW** defines the distance weighting between patch and point with probability 0.05.

Example

```
--model PRESENCE distW=1000,2000
```

The result is stored in the file `model-PRESENCE-dW1000,2000.txt` in the project folder :

Graph	Metric	DistWeight	R2	AIC	Coef
2000m-3500cost	BC_d3500_p0.05_beta1	1000.00	0.229898	56.0689	3.63230e-07
2000m-3500cost	BC_d3500_p0.05_beta1	2000.00	0.134672	62.7547	4.89398e-08
2000m-3500cost	F_d3500_p0.05_beta1	1000.00	0.522162	35.5490	0.00155784
2000m-3500cost	F_d3500_p0.05_beta1	2000.00	0.266793	53.4785	0.000145906
2000m-3500cost	d_PC	1000.00	0.296785	51.3727	471.384
2000m-3500cost	d_PC	2000.00	0.292047	51.7054	460.552

References

[[Foltête et al.\(2012b\)](#), [Foltête et al.\(2012a\)](#), [Girardet et al.\(2013\)](#), [Clauzel et al.\(2013\)](#)]

2.5 Adding/Removal

2.5.1 `--delta`: remove one item

`--delta global_metric_name [maxcost=valcost] [param1=val ...] obj=patch|link`
`[sel=id1,id2,...,idn | fsel=file.txt]`

Required parameters

- **global_metric_name**: global metric short's name (PC, ...)
- **obj=patch|link**: removal of patches (**obj=patch**) or links (**obj=link**)

Optional parameters

- **maxcost=valcost**: limit pathfinding to *valcost* (ie. paths greater than *maxcost* are not calculated). It can reduce the time execution for path-based metric but results may be inaccurate.
- **param1=val**: metric parameter(s), if needed. Ranges are not allowed for this command.
- **sel=id1,id2,...,idn**: restrict the calculation to items (patches or links) listed by identifier
- **fsel=file.txt**: restrict the calculation to items (patches or links) listed in the file *file.txt*. The file must contain one identifier by line.

Description

Calculates global metric in delta mode on patches or links depending on **obj** parameter for each selected graph.

If the **sel** or **fsel** parameter is set, it calculates only on selected items. Results are stored in a text file for each graph in the project folder. The file name is the concatenation of 'delta-' + metric short name + graph name.

Examples

To execute the NC metric in delta mode for each patch:

```
--delta NC obj=patch
```

To execute the PC metric in delta mode for only patch id 2 and 3:

```
--delta PC d=1000 p=0.05 beta=1 obj=patch sel=2,3
```

Results are stored in one file for each graph. One sample:

Id	d_PC
Init	1.3170910074623971E-5
2	2.68908916812349E-3
3	1.738640713802898E-4

Init corresponds to the initial PC value, without removing element. Following values correspond to the relative loss of the metric value when removing the element from the graph. Removing the patch with id equal to 2, the PC decreases by 0.269%

2.5.2 --gremove: remove several items

```
--gremove global_metric_name [maxcost=valcost] [param1=val ...]  
[patch=id1,id2,...,idn|fpatch=file.txt] [link=id1,id2,...,idm|flink=file.txt]
```

Required parameter

- **global_metric_name**: global metric short's name (PC, ...)

Optional parameters

- **maxcost=valcost**: limit pathfinding to *valcost* (ie. paths greater than *maxcost* are not calculated). It can reduce the time execution for path-based metric but results may be inaccurate.
- **param1=val**: metric parameter(s), if needed. Ranges are not allowed for this command.
- **patch=id1,id2,...,idn**: remove the patches listed by identifier

- `fpatch=file.txt`: remove the patches listed in the file *file.txt*. The file must contain one identifier by line.
- `link=id1,id2,...,idn`: remove the links listed by identifier
- `flink=file.txt`: remove the links listed in the file *file.txt*. The file must contain one identifier by line.

Description

Removes listed patches and/or links on each selected graph and calculates the given global metric. The list of identifiers can be given on the command line or in a text file.

Results are only displayed. For each graph, Graphab shows the number of patches and links truly removed. The number of patches does not vary, but the number of links can vary since links connected to a removed patch are also removed. If an identifier does not exist, it will be ignored.

Examples

The following command computes the NC metric on each selected graph after removing patches with id 2 and 3 :

```
--gremove NC patch=2,3
```

Result:

```
Global indice NC
Graph 2000m-3500cost
Remove 2 patches and 5 links
NC : 25.0

Graph 2000m-3500cost_comp
Remove 2 patches and 8 links
NC : 24.0

Graph 2000m_euclid
Remove 2 patches and 7 links
NC : 9.0

Graph 2000m_euclid_comp
Remove 2 patches and 9 links
NC : 9.0
```

The same command can be written as:

```
--gremove NC fpatch=patch.txt
```

With the file patch.txt in the current directory, containing one id by line:

```
2
3
```

2.5.3 -gtest: removal and iterative item addition

```
--gtest nstep global_metric_name [maxcost=valcost] [param1=val ...] obj=patch|link
sel=id1,id2,...,idn | fsel=file.txt
```

Required parameters

- `nstep` : number of items to be added
- `global_metric_name` : global metric short's name (PC, ...)
- `obj=patch|link` : test patches (`obj=patch`) or links (`obj=link`)
- `sel=id1,id2,...,idn` : test items (patches or links) listed by identifier
- `fsel=file.txt` : test items listed in the file *file.txt*. The file must contain one identifier by line.

Optional parameters

- `maxcost=valcost`: limit pathfinding to *valcost* (ie. paths greater than *maxcost* are not calculated). It can reduce the time execution for path-based metric but results may be inaccurate.
- `param1=val`: metric parameter(s), if needed. Ranges are not allowed for this command.

Description

This command removes the items selected by the `sel` ou `fsel` parameter. Then it replaces the removed item which maximizes the metric *global_metric_name*. This process is repeated up to the *nstep* parameter. The whole process is performed for each selected graph. For each graph, the results are stored in 2 text files. The first lists the added element and the corresponding metric value. The second, more detailed, lists, for each step, the adding of each item.

Example

```
--gtest 3 IIC obj=patch sel=1,2,3,5,6,10,15,16
```

This command removes the 8 selected patches and replaces successively the 3 patches which maximize the IIC metric.

For the graph *2000m_euclid*, the first file, named *gtest-2000m_euclid-IIC.txt*, contains:

Step	Id	IIC
0	init	1.7825075712618167E-6
1	1	1.753327449219068E-6
2	15	1.7793775301389374E-6
3	16	1.780788239231567E-6

The first line (Step 0) corresponds to the metric value before removing the items. The following lines shows, for each step, the replaced patch and the metric value after adding this patch.

For the same graph, the detailed file, named *gtest-2000m_euclid-IIC-detail.txt*, contains:

Step	Id	IIC
1	init	1.6852479916976776E-6
1	16	1.6866587007903073E-6
1	1	1.753327449219068E-6
1	2	1.685330641324248E-6
1	3	1.6852916014495828E-6
1	5	1.6858693558888445E-6
1	6	1.6852486177082585E-6
1	10	1.6854627861279089E-6
1	15	1.6994856702980223E-6
2	init	1.7533274492190677E-6
2	16	1.7547381583116972E-6
2	2	1.753410098845639E-6
2	3	1.7538298458957464E-6

2	5	1.753948813410235E-6
2	6	1.7533280752296487E-6
2	10	1.753542243649299E-6
2	15	1.7793775301389374E-6
3	init	1.7793775301389372E-6
3	16	1.780788239231567E-6
3	2	1.7794601797655085E-6
3	3	1.7799512085537507E-6
3	5	1.7799988943301041E-6
3	6	1.7795590940743919E-6
3	10	1.7795923245691686E-6

2.5.4 –addpatch : iterative patch addition

```
--addpatch npatch global_metric_name [param1=val ...]
[gridres=min:inc:max [capa=capa_file] [multi=npatch,size]]
| [pointfile=file.shp [capa=capa_field]]
```

Global paramaters

- *npatch* : number of patches to be added
- *global_metric_name* : global metric short's name (PC, ...)
- *param1=val*: metric parameter(s), if needed. Ranges are not allowed for this command.

Description

This command tests successively the adding of a new patch from a predefined set of points and retains the patch which maximizes the given global metric. This process is repeated as many times as the *npatch* parameter and for each selected graph. Results are stored in a project subdirectory created for each graph.

For each testing of a new patch, the graph is recomputed including the new patch and the links connecting this patch to the others, then the given metric is calculated on this new graph. When all the possible patches was tested, the one maximizing the metric is added to the project. This process is iterated until having *npatch* new patches in the project.

The created patches have a size of one pixel, if the corresponding land cover pixel is already habitat or is outside of the study area, the patch is skipped. Patches location to be tested can be given through a regular grid or a set of points from a shapefile.

Point set test

```
--addpatch npatch global_metric_name [param1=val ...] pointfile=file.shp [capa=capa_field]
```

For each point of the shapefile, the program tests the addition of a patch on the pixel covering the point, if the pixel is not NoData or already in the habitat class.

The *capa* parameter defines an attribute of the shapefile containing a capacity value for each tested patch. If the *capa* parameter is not specified, the capacity of new patches will be 1.

Regular grid test

```
--addpatch npatch global_metric_name [param1=val ...] gridres=min:inc:max [capa=capa_file]
[multi=npatch,size]
```

Tests patches addition on a regular grid. The size of the grid cell is set by **gridres**.

The **capa** parameter is used to define a raster file (TIFF or AsciiGrid format) giving a value of potential capacity at any point of the study area. If the capacity is zero, no patch will be tested at this position. The raster file can have a different resolution than the grid or the landscape map. If the **capa** parameter is not specified, the capacity of new patches will be 1.

The **multi** parameter is used to test the simultaneous addition of *nbpatch* patches, in a neighborhood of radius *size*gridres*.

Examples

```
--addpatch 5 IIC gridres=100
```

Adds 5 patches maximizing metric IIC, testing the addition of a patch every 100 meters. Results are stored in *addpatch_n5_graph_IIC_res100_multi1_1* subdirectory :

- *addpatch_graph_IIC.shp* : contains the added patches and the metric value
- *addpatch_graph_IIC.txt* : contains for each added patch the metric value
- *links_graph_IIC.shp* : contains the linkset of the final graph
- *topo-links_graph_IIC.shp* : contains the topological linkset of the final graph
- *detail/* : subdirectory containing the detail of each test for each step
- *detail/detail.i.shp* : tested points set for the addition of the i-th patch

The command line below is equivalent to the previous one but will run at 3 different resolutions (100 200 500). The results will be stored in three folders, one for each resolution.

```
--addpatch 5 IIC gridres=100,200,500
```

Another example with a shapefile points set :

```
--addpatch 5 IIC pointfile=testpoint.shp
```

Results are stored in *addpatch_n5_graph_IIC_shptestpoint.shp* subfolder.

Limitations

The **--addpatch** command only works with graphs from complete topology linkset.

This command changes the number of patches in the project; executing commands after it can lead to inconsistencies. Therefore, it is not recommended to add commands after the command **--addpatch**.

References

[[Clauzel et al.\(2015a\)](#), [Foltête et al.\(2014\)](#)]

2.5.5 --remelem: iterative item removal

```
--remelem nstep global_metric_name [maxcost=valcost] [param1=val ...] obj=patch|link  
[sel=id1,id2,...,idn | fsel=file.txt]
```

Required parameters

- **nstep** : number of items to be removed
- **global_metric_name** : global metric short's name (PC, ...)
- **obj=patch|link** : remove patches (obj=patch) or links (obj=link)

Optional parameters

- `sel=id1,id2,...,idn` : test items (patches or links) listed by identifier
- `fsel=file.txt` : test items listed in the file *file.txt*. The file must contain one identifier by line.
- `maxcost=valcost`: limit pathfinding to *valcost* (ie. paths greater than *maxcost* are not calculated). It can reduce the time execution for path-based metric but results may be inaccurate.
- `param1=val`: metric parameter(s), if needed. Ranges are not allowed for this command.

Description

The `--remelem` command tests the removing of graph items (patches or links according to the `obj` parameter) like the `--delta` command. Then it removes the item which minimizes the given metric *global_metric_name*. This process is repeated up to *nstep*. If the `sel` or `fsel` parameter is set, the test is limited to the selected items. The whole process is performed for each selected graph. For each graph, the result is stored in a text file that lists the removed items and the corresponding metric value.

Example

```
--remelem 3 IIC obj=patch
```

This command successively removes the 3 patches which minimize the IIC metric

For the graph *2000m_euclid*, the file, named *rempatch-IIC-2000m.euclid.txt* contains:

Step	Id	IIC
0	init	1.7825075712618167E-6
1	120	1.070922766241333E-6
2	145	8.001250544646545E-7
3	118	6.05326129866607E-7

The first line (Step 0) corresponds to the initial value of the metric. The following lines show, for each step, the removed item and the metric value after removing.

Reference

[Foltête et al.(2016)]

2.6 Land use changes

2.6.1 --landmod : land use changes

```
--landmod zone=filezones.shp id=fieldname code=fieldname [selid=id1,...,idn] [novoronoi]
```

Required parameters

- `zone=filezones.shp` : polygon shapefile containing land use changes.
- `id=fieldname` : field name of the shapefile used for identifying polygons. If values are not unique, polygons with the same identifier will be applied in a single change.
- `code=fieldname` : field name of the shapefile storing the new land use category of the polygon.

Optional parameter

- **selid=id1,...,idn** : list of polygon identifiers to process. If this parameter is not set, all polygons of the shapefile are processed.
- **novoronoi** : at project creation, do not calculate the planar topology. Useful to speedup execution when planar topology is not used.

Description

This command must be placed before other calculation commands. It will duplicate the project for each polygon of the shapefile and change the landuse covered by the polygon. The commands following this one, will be executed on each modified project. The newly created projects does not contain the linksets and graphs of the initial project.

The newly created projects will be named by the identifier of the polygon(s) and stored in the project directory.

Example

```
--landmod zone=zones.shp id=ID code=CODE --linkset distance=euclid --graph
--gmetric IIC
```

For each polygon identifier contained in the **zones.shp** shapefile, a new project named with the ID value will be created in a sub directory of the current project. On each project, a linkset in euclidean distance and a graph will be created and the IIC metric will be calculated on the graph. Finally, each subdirectory of the current project will contain a file **IIC.txt** containing the metric value taking into account each land use changes.

Reference

[[Sahraoui et al.\(2017\)](#)]

2.7 Options

2.7.1 -nosave

This option prevents saving a project. It is useful when you don't want commands to modify the project, like **--linkset**, **--graph**, **--pointset**, **--cmetric**, **--lmetric**...

2.7.2 -proc

Defines the number of processors (or cores) used by Graphab. By default, CLI mode uses the value defined in the preferences window. See the paralellism section for more details.

2.7.3 -mpi

This option is used to execute commands on several computers in the MPI environment. See the paralellism section for more details.

Chapter 3

Command examples

All the following examples can be tested with the sample project available on Graphab website.

3.1 Display project

First, display project elements :

```
java -jar graphab-2.2.jar --project sample_project/Project.xml --show
```

Result :

```
===== Link sets =====
```

```
Complete_Euclid
```

```
Complete_cost
```

```
Planar_Euclid
```

```
Planar_cost
```

```
===== Graphs =====
```

```
2000m-3500cost
```

```
2000m-3500cost_comp
```

```
2000m_euclid
```

```
2000m_euclid_comp
```

```
===== Point sets =====
```

```
Presence_absence
```

3.2 Batch metric parameter

Calculates PC metric on the graph *2000m-3500cost* varying *d* parameter from 1000 to 5000 by step of 1000 :

```
java -jar graphab-2.2.jar --project sample_project/Project.xml  
--usegraph 2000m-3500cost --gmetric PC d=1000:1000:5000 p=0.05 beta=1
```

Results are written in the file PC.txt in the project folder :

Graph	d	p	beta	PC	
2000m-3500cost	1000.0	0.05	1.0	1.3170910074623973E-6	
2000m-3500cost	2000.0	0.05	1.0	1.5884005111333572E-6	
2000m-3500cost	3000.0	0.05	1.0	1.7406772135728036E-6	
2000m-3500cost	4000.0	0.05	1.0	1.8425812214129719E-6	
2000m-3500cost	5000.0	0.05	1.0	1.918332024845314E-6	

3.3 Batch thresholded graph and global metric

Creates 6 thresholded graphs from the linkset *Complete_cost* and computes the IIC metric :

```
java -jar graphab-2.2.jar --project sample_project/Project.xml --uselinkset Complete_cost
--graph threshold=2000:100:2500 --gmetric IIC
```

Results are written in the file IIC.txt in the project folder :

Graph	IIC
thresh_2000.0_Complete_cost	1.3740319506984741E-6
thresh_2100.0_Complete_cost	1.3742497768764619E-6
thresh_2200.0_Complete_cost	1.3749834046381206E-6
thresh_2300.0_Complete_cost	1.3754601882078134E-6
thresh_2400.0_Complete_cost	1.3857662689627643E-6
thresh_2500.0_Complete_cost	1.4197576370824857E-6

3.4 Complete SDM sequence

Calculates a graph from the linkset *Planar_Euclid*, calculates 2 metrics (Dg and F) on the created graph, adds pointset on *Planar_Euclid* linkset and calculates SDM for the two metrics versus the PRESENCE variable, without modifying the project.

```
java -jar graphab-2.2.jar -nosave --project sample_project/Project.xml
--uselinkset Planar_Euclid
--graph
--lmetric Dg --lmetric F d=1000 p=0.05 beta=1
--pointset sample_project/Exo-Presence_absence.shp
--model PRESENCE distW=1000
```

Results are stored in the file model-PRESENCE-dW1000.txt in the project folder :

Graph	Metric	DistWeight	R2	AIC	Coef
comp_Planar_Euclid	Dg	1000.00	0.999744	2.01795	24.8994
comp_Planar_Euclid	F_d1000_p0.05_beta1	1000.00	0.108353	64.6026	2.38405e-05

Chapter 4

Performance tuning

4.1 Parallelism to speed up execution

4.1.1 One computer : threads

If your computer has more than one core (most of them), you can take advantage of parallelization. Most Graphab commands are parallelized. You can speed up command execution by defining the number of cores (or processors) used by Graphab with the option `-proc` after the project command :

```
java -jar graphab-2.2.jar -proc 8 --project path2myproject/myproject.xml ...
```

By default, CLI mode uses the number of processors defined in the preferences window of the GUI.

4.1.2 Computer cluster : mpi

Graphab can be run on computer clusters wich support Java for OpenMPI.

```
mpirun java -jar graphab-2.2.jar -mpi --project path2myproject/myproject.xml ...
```

Only some commands can be used in mpi environments : `--gmetric`, `--cmetric`, `--lmetric`, `--delta`, `--addpatch`, `--gtest`, `--remelem`, `--landmod`

4.2 Memory management

In CLI mode, the memory configuration defined in the preferences window cannot be used. By default, the amount of memory available for Graphab is system dependent. It can vary from 128 Mb to several Gb. In most cases, Graphab will run normally. But if you have a large project, some commands would be slow or even crash due to memory limitation. If Graphab execution terminates with `OutOfMemoryError` or GC overhead, you need to increase memory allocated to Graphab.

To define manually the maximum amount of memory allocated to Graphab, use Java option `-Xmx`:

```
java -Xmx4g -jar graphab-2.2.jar ... # 4Gb allocated
java -Xmx1500m -jar graphab-2.2.jar ... # 1500 Mb -> 1.5Gb allocated
```

If you cannot allocate more than 1Gb or 1.5G and your computer has more memory available, you have probably a 32-bit version of Java, which is limited to less than 2Gb of memory. Check your Java version :

```
java -version
```

If it is a 32-bit version, install a 64-bit Java version to handle all your computer memory.

Bibliography

- [Clauzel et al.(2015a)] Celine Clauzel, Cyrielle Bannwarth, and Jean-Christophe Foltete, 2015a. Integrating regional-scale connectivity in habitat restoration: An application for amphibian conservation in eastern france. *Journal for Nature Conservation*, 23 98 – 107.
- [Clauzel et al.(2015b)] Celine Clauzel, Deng Xiqing, Wu Gongsheng, Patrick Giraudoux, and Li Li, 2015b. Assessing the impact of road developments on connectivity across multiple scales: Application to yunnan snub-nosed monkey conservation. *Biological Conservation*, 192 207 – 217.
- [Clauzel et al.(2013)] Céline Clauzel, Xavier Girardet, and Jean-Christophe Foltête, 2013. Impact assessment of a high-speed railway line on species distribution: Application to the european tree frog (*hyla arborea*) in franche-comté. *Journal of Environmental Management*, 127 125 – 134.
- [Foltête and Vuidel(2017)] Jean-Christophe Foltête and Gilles Vuidel, 2017. Using landscape graphs to delineate ecologically functional areas. *Landscape Ecology*, 32(2) 249–263.
- [Foltête et al.(2012a)] Jean-Christophe Foltête, Céline Clauzel, and Gilles Vuidel, 2012a. A software tool dedicated to the modelling of landscape networks. *Environmental Modelling and Software*, 38 316 – 327.
- [Foltête et al.(2012b)] Jean-Christophe Foltête, Céline Clauzel, Gilles Vuidel, and Pierline Tournant, 2012b. Integrating graph-based connectivity metrics into species distribution models. *Landscape Ecology*, 27(4) 557–569.
- [Foltête et al.(2016)] Jean-Christophe Foltête, Geoffroy Couval, Marilyne Fontanier, Gilles Vuidel, and Patrick Giraudoux, 2016. A graph-based approach to defend agro-ecological systems against water vole outbreaks. *Ecological Indicators*, 71 87 – 98.
- [Foltête et al.(2014)] Jean-Christophe Foltête, Xavier Girardet, and Céline Clauzel, 2014. A methodological framework for the use of landscape graphs in land-use planning. *Landscape and Urban Planning*, 124 140 – 150.
- [Girardet et al.(2013)] Xavier Girardet, Jean-Christophe Foltête, and Céline Clauzel, 2013. Designing a graph-based approach to landscape ecological assessment of linear infrastructures. *Environmental Impact Assessment Review*, 42 10 – 17.
- [Newman(2006)] M. E. J. Newman, 2006. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23) 8577–8582.
- [Sahraoui et al.(2017)] Yohan Sahraoui, Jean-Christophe Foltête, and Céline Clauzel, 2017. A multi-species approach for assessing the impact of land-cover changes on landscape connectivity. *Landscape Ecology*, 32(9) 1819–1835.