Graphab 3.0

User Manual

Marc Bourgeois, Céline Clauzel, Jean-Christophe Foltête, Xavier Girardet, Paul Savary, Gilles Vuidel

2024-12-02

Contents

Chapter 1

Introduction

1.1 About Graphab

Graphab is a software application for modeling ecological networks using landscape graphs. It is composed of four modules for:

- constructing graphs, including loading initial landscape data and identifying patches and links (Euclidean distances or least-cost paths)
- computing connectivity metrics from graphs
- linking graph-based connectivity metrics with external data
- visual and cartographic interfacing

A QGIS plugin (Graphab4QGIS) lets you use Graphab directly from QGIS:
https://plugins.qgis.org/plugins/graphab4qgis/

1.1.1 Authors

Graphab has been developed by Gilles Vuidel and Jean-Christophe Foltête at ThéMA laboratory (University of Franche-Comté – CNRS) with contributions from Anissa Bellil, Marc Bourgeois, Céline Clauzel, Stéphane Garnier, Xavier Girardet, François-Marie Martin, Yohan Sahraoui and Paul Savary. 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 one of the following references [Foltête et al.(2012a), Foltête et al.(2021)] in their publications:
Foltête J.-C., Vuidel G., Savary P., Clauzel C., Sahraoui Y., Girardet X., Bourgeois M. 2021. Graphab: an application for modeling and managing ecological habitat networks. Software Impacts.8: 100065.
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 11 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 Processing capabilities and limitations below and the journal article [Foltête et al.(2012a)].

1.3 Installing the software and launching a project

Graphab can be downloaded from `https://sourcesup.renater.fr/www/graphab`.

- Download and install Java 11 or later - adoptium.net. It is best to install the 64-bit version of Java.
- Download graphab-3.0.jar
- Launch graphab-3.0.jar

Please note that projects are not compatible between Graphab versions 2.x and 3.x !

Graphab can be used through 2 user interface : the graphical user interface (GUI) and the command line interface (CLI).

The graphical user interface allows to create project, visualize the data, calculate metrics, etc. The command line interface allows to run the same calculations on remote computers as well as on computer clusters.

The chapter 2 describes all of the software features available from the GUI. Chapters 3, 4, 5 details how to use Graphab on the command line. The set of available metrics is detailed in Chapter 7. And Chapter 6 reviews the parallelization modes, performance and memory management.

Chapter 2

Graphical user interface

After launching Graphab, the File menu provides access to four sections:

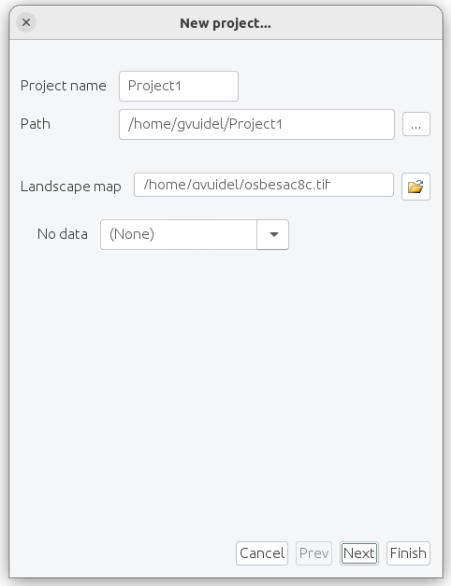
- File / New project: to create a new project in which all data and results are saved automatically.
- File / Open project: to open an existing project.
- File / Preferences: to change certain software parameters: English/French; maximum amount of memory to use; number of processors to use. It is recommended to adjust the amount of memory and number of processors to suit your computer (see section Processing capabilities and limitations).
- File / Log window: to display the event log.

2.1 Starting a Graphab project

New projects are created from the `File / New project` menu. The user must complete a series of windows to identify the project, import a landscape map, and create a link set. Each project is associated with a single landscape map but may contain several habitats, link sets and graphs. After the start phase, the project is the medium for creating multiple graphs and for computing connectivity metrics.

2.1.1 Identifying a project

In the first window, the user must enter a project name and specify the folder in which it is to be created.



The second element is for importing the landscape map. It must be a raster file (*.tif, *.asc, *.rst) in which the value of each pixel corresponds to a category (land cover or other classification).

If the raster format is *.tif without a Geotiff extension, the file must be associated with a world file for geolocation (*.tfw) structured as follows:

Example	
10.0	Pixel size in the X-direction
0.0	Rotation about X-axis
0.0	Rotation about Y-axis
-10.0	Pixel size in the Y-direction
821755.0	X coordinate of the center of the upper-left pixel
2342995.0	Y coordinate of the center of the upper-left pixel

If the raster format is *.rst, the file must be associated with a georeferencing file (*.rdc) generated by Idrisi software.

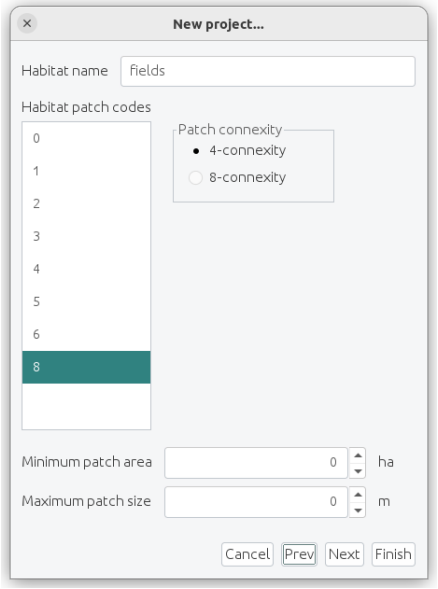
The units of the image coordinate system must be meters. If not, the areal and distance units will be incorrect. The image can be reprojected in a metric projection (UTM, Lambert93) using GIS software.

`No data`: pixel value representing the absence of data in the raster file.

Based on this information, the project can be created by clicking on the `Finish` button, or you can continue with the habitat definition by clicking on the `Next` button.

2.1.2 Creating habitat

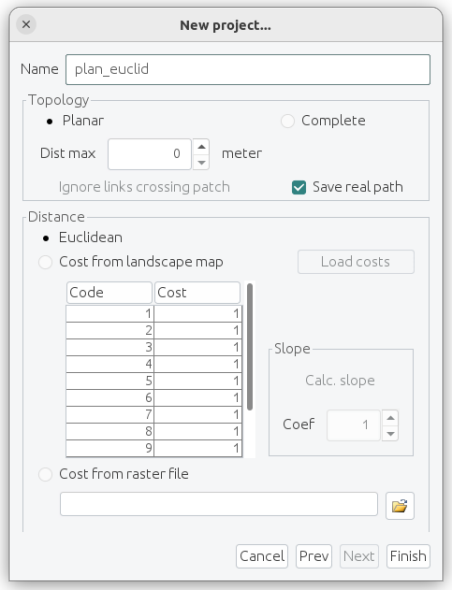
From the landscape map, the habitat can be defined on the basis of several parameters detailed in section Creating raster habitat.



Based on this information, the project can be created with an habitat by clicking on the `Finish` button, or you can continue with the linkset definition by clicking on the `Next` button.

2.1.3 Creating link set

The third window is for creating a link set based on the habitat defined above. The creation of this link set is the final step in the initial set-up of the project. However, new link sets can be created at any time within the same project (cf. Link set).



It's often simpler and quicker to start with a planar link set in euclidean distance (default settings).

Based on this information, the project can be created with a habitat and a link set by clicking on the `Finish` button.

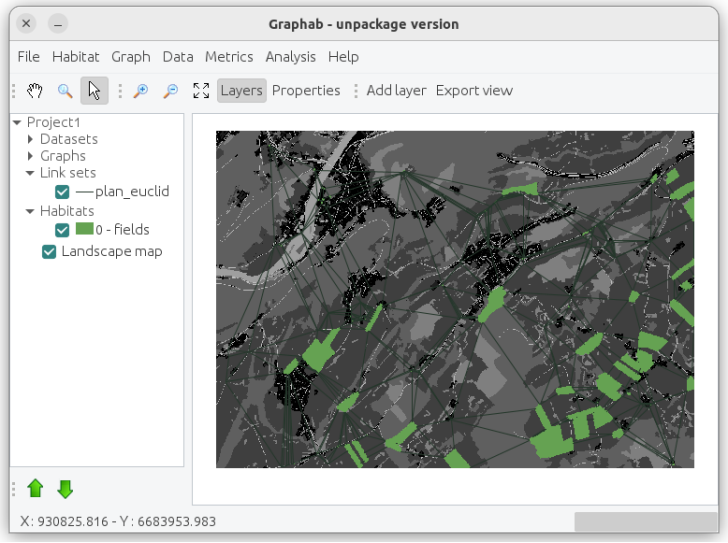
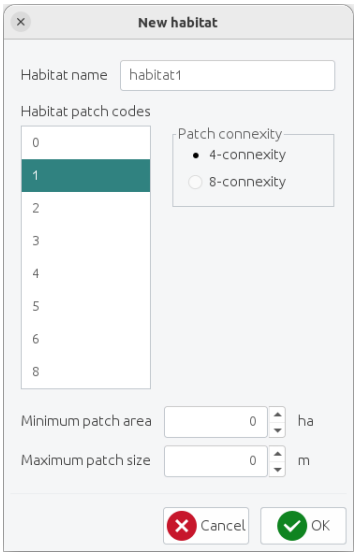


Figure 2.1: The new created project with the landscape map, the habitat and the linkset shown

2.2 Habitat

2.2.1 Creating raster habitat

To create a habitat from the landscape map, use the `Habitat / Add raster habitat` menu.



From the landscape map, the habitat can be defined with the following parameters:

`Habitat name`: name of the new habitat; avoid spaces and special characters. It must be unique within the project.

`Habitat patch codes`: pixel values assigned to the habitat category used to define habitat patches. Multiple values can be selected by holding down the `Ctrl` key.

`Minimum patch area`: minimum area in hectares for a habitat patch to become a graph node.

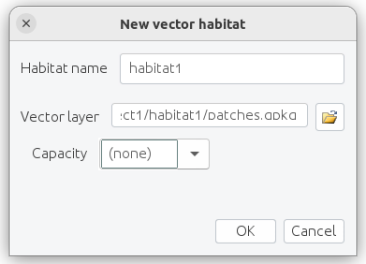
`Maximum patch size`: length in meters beyond which the patch is split into several pieces. This option is useful for avoiding habitat patches covering too large area.

`Patch connexity`:

- 4-connexity: a habitat patch consists of the central pixel with its four neighbors if they are of the same value;
- 8-connexity: a habitat patch consists of the central pixel with its eight neighbors if they are of the same value.

2.2.2 Creating vector habitat

It is also possible to create a habitat from a vector layer, independent of the landscape map, using the `Habitat / Add vector habitat` menu.



Parameters

`Habitat name`: name of new habitat; avoid spaces and special characters.

`Vector layer`: vector-format layer containing habitat patches. The layer's extent must be included in the landscape map's extent.

`Capacity`: layer attribute providing information on the capacity of each patch. This parameter is optional and can be left at `(None)`, in which case the capacity will correspond to the surface area in m^2 of the patches.

2.2.3 Patch capacity

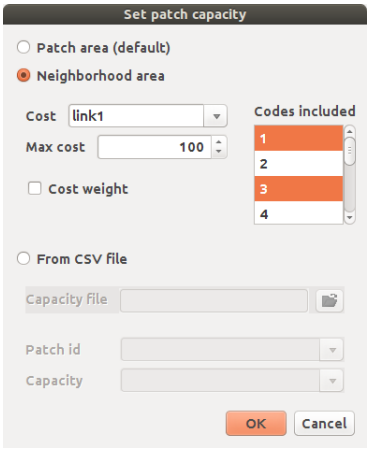
The capacity of a patch reflects its intrinsic quality as an indicator of its demographic potential. A patch with a high capacity can accommodate a large population and vice versa. Capacity is included directly in the calculation of some area connectivity metrics and weighted connectivity metrics (see Metrics).

By default (when the habitat is created), the patch capacity is equal to the patch area in m^2. However, users may replace area by any other quality indicator. In some cases, species presence is related not to patch size but to the area of other types of land cover around the patch. For example, the presence of amphibians in a breeding pond does not depend on the pond size but on the amount of terrestrial habitat surrounding the pond.

The `Habitat / Set patch capacity` menu can be used to modify the capacity of the patches by two methods.

Capacity as a function of the neighborhood

The first method can be used to define patch capacity as a function of the neighborhood composition and to calculate it directly from Graphab.



Users must define three parameters: type of distance, maximum distance, landscape categories.

`Cost`: This is the spatial metric (Euclidean or cost distance) corresponding to one link set available in the project. The use of costs in this procedure amounts to defining an anisotropic neighborhood around patches which may differ greatly from a buffer function. For consistency, it is recommended to use the same type of distance as was used in creating the links of the graph.

`Max cost`: Like the graph maximum distance, the unit of this maximum distance depends on the type of distance used in creating the link set (Euclidean or cost distance).

`Codes included`: the user may select one or more landscape categories, other than the habitat category, to be included in calculating capacity.

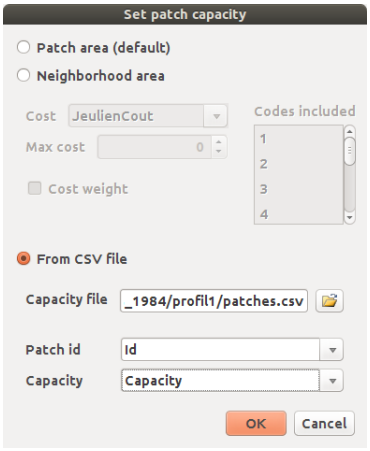
The `cost weight` option introduces a weighting with distance to the patch through a negative exponential function. In this way, the areas selected have greater weight if they are close to the patch and vice versa.

The capacity values calculated replace the patch area for all subsequent computations. But users can return to the initial parameter by selecting `Patch area`.

Capacity defined from external data

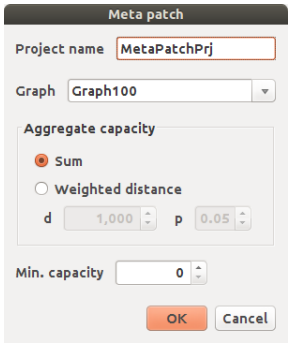
The second method is for importing a data table (*.csv) describing all the patches of the project and containing capacity values defined in advance by the user. The patch identifiers in the table must be the same as the patch identifiers in the Graphab project.

The capacity values in the imported table replace patch area values for all subsequent computations after importing. But users can restore the initial parameter by selecting `Patch area`.



2.2.4 Meta-patch

From the Habitat menu, the `Create metapatch project` entry creates a new habitat where the nodes of the graph correspond to sets of interconnected patches. This feature allows to better take into account the nested scales of ecological processes, especially the functional definition of a habitat patch that is dependent on the scale considered (Theobald, 2006 ; Zetterberg et al., 2010).



To use this feature, it is necessary to create an initial project and a graph pruned at the desired distance (eg daily distance). Each component of this pruned graph correspond to a meta-patch. It is therefore constituted by all interconnected patches.

By default, the metapatch capacity is defined as the sum of the capacity of the patches composing the metapatch. The other option, `Weighted distance`, is used to reflect the removal between patches. For this option, you must define the parameter α by giving a distance d and an associated probability p (see Parameters of weighted metrics). The resulting capacity (C) of a metapatch is:

$$C = \frac{1}{n} \sum_{i}^{n} \sum_{j}^{n} c_j e^{-\alpha d_{ij}}$$

The `Min. capacity` option keeps only the metapatch that have a capacity greater than or equal to the given capacity.

The new habitat is saved into the project and is automatically shown at the end of the process.

Reference : [Clauzel et al.(2015b)]

2.2.5 Additional functions (context menu)

Certain functions can be accessed via the context menu for each habitat in the project tree (right-click on the name of a habitat).

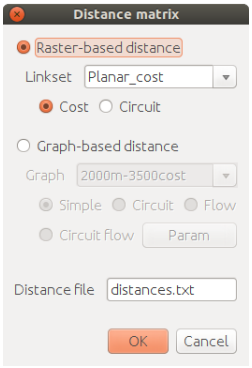
Remove patches

After defining a different capacity for the surface area of the patch, it may be useful to remove patches whose capacity is considered to be too low to meet the needs of the species being studied. The `Remove patches` entry in the contextual menu is used to create a new habitat identical to the selected habitat by removing patches with a capacity below a given threshold.

Once the required minimum capacity has been entered, a new habitat is created in the project with the same name as the original habitat, suffixed with the minimum capacity threshold.

Matrix of inter-patch distances

The `Distance matrix` entry in a habitat's context menu can be used to calculate different distances between all the patches in the habitat. This function is useful, for example, for comparing these distances with genetic distances.



Distances can be calculated in 2 ways: directly on the raster or via a graph.

Raster distance In raster mode, distances are calculated according to a reference link set. If this link set is not in Euclidean distance, the calculation incorporates the costs associated with the landscape map categories, as defined in the set of links. The result is a distance matrix independent of the graph; this matrix corresponds to the calculation offered by Geographic Information Systems.

The `Circuit` option calculates the 'resistance distance' between each point, i.e. the equivalent resistance of a network of electrical resistors. This method allows all possible paths to be taken into account, not just the shortest (see [McRae et al.(2008)]).

Distance on graph In graph mode, distances are calculated according to the selected graph.

- `Cost`: distance calculated along the shortest path on the reference graph. The distance type corresponds to that used to define the set of links used to define the reference graph. Depending on the choice made when the graph was created, the calculation may include intra-path distances.
- `Circuit`: calculates the "resistance distance" between the nodes of the graph. The graph is seen as an electrical network where each link in the graph represents a resistance. This option allows all possible paths to be taken into account, not just the shortest.
- `Flux`: functionality under test
- `Circuit flow` : functionality under test

Removing a habitat

The `Remove` entry is used to delete the selected habitat. The project is automatically saved after this deletion. All link sets, graphs and metrics calculated from this habitat are automatically deleted !

Habitat properties

The `Properties` entry recalls the parameters used to build the habitat, as well as the number of patches.

2.3 Link set

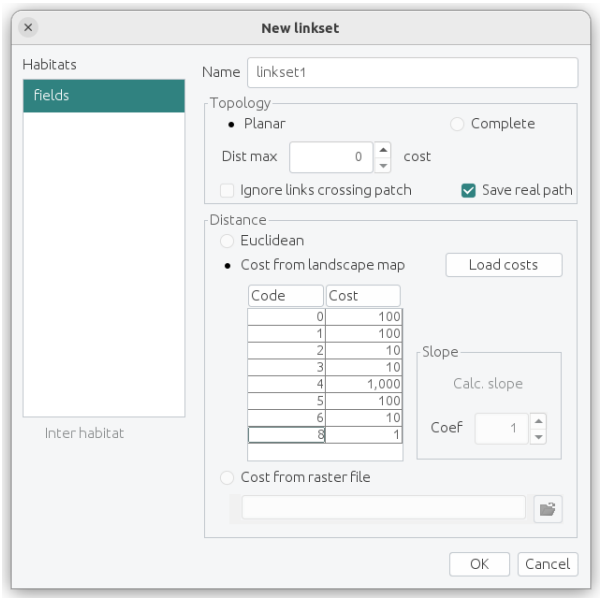
A link set corresponds to paths linking habitat patches together. It depends on one or more habitats.

2.3.1 Creating a link set

A link set is created from the menu `Graph / Create link set`.

The first element to define is the habitat(s) to be included in this link set. To select several habitats, use the `Ctrl` key.

`Name`: name of the new link set; avoid spaces and special characters. It must be unique in the project.



Link topology

Two topologies are available:

- planar: only links that form a minimal planar graph are considered. This topology is set up through Voronoi polygons around each habitat patch. These polygons are defined from the edges of patches in Euclidean distance.
- complete: all the links between patches are potentially taken into account.

`Inter habitat` : dans le cas d'un jeu de liens multi-habitat, la case à cocher `Inter habitat` permet de modifier la topologie en conservant uniquement les liens entre tache d'habitats différents.
Attention, cette option doit être utilisée normalement avec une topologie complète !

`Dist max`: this option specifies a threshold distance. Links that exceed this distance are no longer created. This limits the number of links created and so accelerates the creation of the link set. The unit of the distance depends on the type of distance: in meters for the euclidean distance and in cost for the least cost distance.

`Ignore links crossing patch`: this option means that a link between two patches (A and C in the figure below) which crosses an intermediate patch (B) is not created. It is recommended for calculating the betweenness centrality metric (BC) to take into account how often a patch lies on the shortest path between all pairs of patches in the graph. If the option is unchecked, a link is created between two patches (A and C) crossing an intermediate patch, representing the complete true distance between A and C.

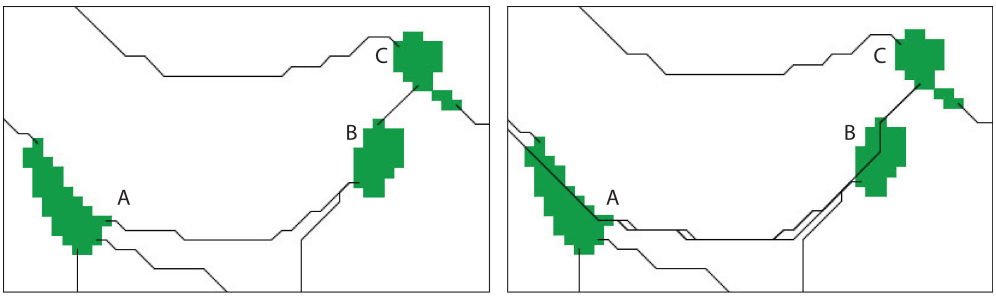


Figure 2.2: Illustration of the option Ignore links crossing patch

`Save real path`:

- checked: links are saved as paths representing the actual route of the link between two patches.
- unchecked: links are saved in topological form only. In this case, display of links with the realistic view is unavailable. This is recommended for graphs with very many links (e.g. a non-thresholded complete graph) so as to limit the use of memory. Unless paths are saved, intra-patch distances cannot be included in the computation of metrics.

Distance (or link impedance)

Distances are calculated from edge to edge between patches. Two main types of distance are available: Euclidean distances and least-cost distances.

- Euclidean distance: links are defined in Euclidean distances (distance as the crow flies between patches), meaning the matrix is considered to be uniform.
- Least-cost distance: links are defined in cost distances. This distance is equal to the sum of the costs of all the pixels along the least-cost path. Matrix heterogeneity is taken into account by assigning a resistance value (friction) to each landscape category. The user can activate this option in either of two ways:
  1. either by specifying different costs for the landscape map categories in the table,
  2. or from an external raster file (*.tif, *.asc or *.rst) in which each pixel has a resistance value.

`Load costs`: by clicking on this button, the costs defined for another link set in this project or another one can be loaded automatically, avoiding the need to enter each cost values.

For each link created, its metric length (`DistM`) and its cost-unit distance (`Dist`) are saved and available in "link properties" (see Properties of map elements).

Slope Resistance values previously defined can be weighted by slope [Clauzel et al.(2015b)]. To compute slope, the DEM should be imported from the menu Data | Set DEM raster.

The parameter coef (c) can adjust the importance of the slope (p) weighting. For a given pixel, the resulting resistance value (r_{final}) is computed given the formula:

$$r_{final} = r * (1 + c.p)$$

with the slope $p = \frac{h}{l}$ ratio of the height (h) and the length (l).
$p = 0$ if the slope is flat, $p = 1$ for a slope of 100% ($h = l$).

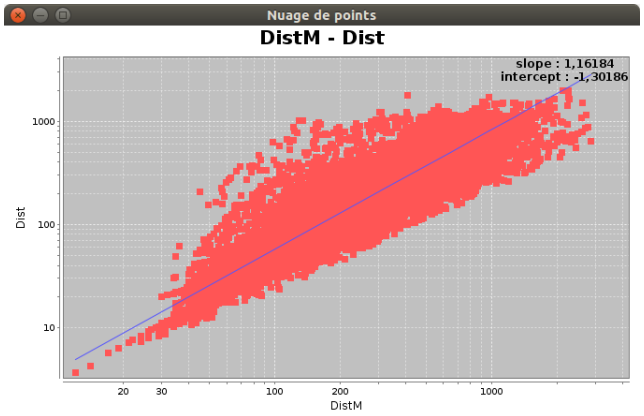
When $c = 1$ the resistance value is doubled for a slope of 100%.
When $c = 10$ the resistance value is doubled for a slope of 10%.

2.3.2 Distance conversion

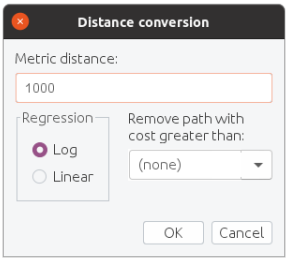
In the case of a link set in cost distance, it is often useful to convert a metric distance into its equivalent in cost distance, in order to prune the graph or set the distance parameters for a metric.

An approximation of the distance metric (DistM) expressed as a cumulative cost (Dist) can be obtained by displaying the double log plot of the link set and using the equation below to perform the conversion:

$$Dist = e^{slope.log(DistM)}$$



The estimation can be performed directly in Graphab from the menu `Distance conversion` available by right-clicking on a link set name.



The parameter `Remove path with cost greater than` allows to exclude from the regression the paths crossing pixels with cost greater or equal to the given value. It is useful to exclude links crossing barrier elements from the distance conversion. This parameter cannot be used with linkset using external cost raster.

2.3.3 Additional functions (context menu)

Certain functions can be accessed via the context menu for each link set in the project tree (right-click on the name of a link set).

Distance conversion

The `Distance conversion` entry is used to convert a metric distance into a cost distance cf. Distance conversion.

Cost extraction

Functionality under test.

Removing a link set

The `Remove` entry is used to remove the selected link set. The project is automatically saved after this deletion. All graphs and metrics calculated from this link set are automatically deleted!

Link set properties

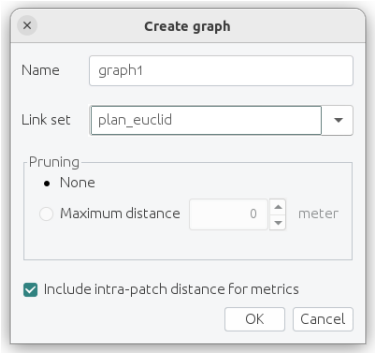
The `Properties` entry lists the parameters used to build the link set and the number of links it contains.

2.4 Graphs

2.4.1 Creating graphs

A Graphab project may entail the creation of several graphs. Each graph is created from a given link set: either the link set defined in the initial project, or a new link set defined from the `Graph / Create link set` menu.

Graphs are created from the `Graph / Create graph` menu.



First, the new graph must be named. Spaces and special characters should be avoided, and the name should be unique within the project.

The user must select one of the link sets created previously (cf. Link set). A graph can only be based on a single link set. To create a graph based on several link sets, see Merge graphs.

Pruning You can choose whether or not to prune the graph:

- None: all links between patches are validated, regardless of length.
- Maximum distance: the selected links are less than or equal to the selected maximum distance.

For a pruned graph with maximum distance, the unit of the distance depends on the type of distance used in creating the link set. If the link set is created using Euclidean distances, the maximum distance is in meters. If the link set is created using cost distances, the maximum distance is given as a cumulative cost. The unit of the distance (cost or meter) is shown just after the `Maximum distance` field.

It is possible to obtain an approximation of the metric distance expressed in cumulative costs from the `Distance conversion` menu, present in the contextual menu of a link set cf. Distance conversion.

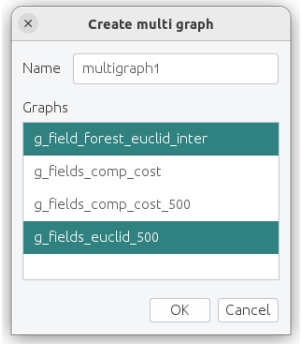
To perform a multiscale analysis, it is often necessary to create a series of graphs in which increasing maximum distances are defined. Users can create this series manually. But if the objective is to analyze the behavior

of a metric according to the maximum distance, the `Metrics / Batch graph` menu can be used (see Batch graph).

`Include intra-patch distances for metrics` option: if the box is checked, the computation of metrics includes the distances between and across patches (recommended). If the box is unchecked, only the distances between (but not across) patches are taken into account.

2.4.2 Merge graphs

Graphab lets you merge several graphs into a single one. The main advantage is that you can group several link sets into a single graph. This function can be accessed via the menu `Graph / Merge graphs`.



Name the new graph and select the graphs to be merged using the `Ctrl` key.

2.4.3 Corridors

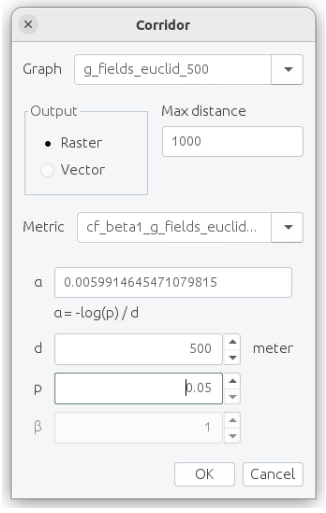
Graphab allows to calculate the corridors representing, for a given maximum distance d_{max}, the area that can be traversed between 2 patches habitat *ie.* the area representing the set of possible paths connecting 2 patches and having a distance less than the distance d_{max}.

This feature can be accessed from the `Graph / Corridor` menu.

Corridors are calculated using a graph to define costs and links.

The maximum cost distance d_{max} must be entered before starting the calculation.

If the graph is defined in Euclidean distance, all costs are defined at the resolution of the landscape map.



Depending on the output type (raster or vector), the settings are different.

Raster output

In the case of raster output, several additional parameters must be entered: the distance decay parameter α from a distance d and a probability p (see Parameters of weighted metrics), and a local metric calculated on the links of the graph.

Whatever the chosen settings, the general principle is the same. A probability gradient is calculated between each pair of linked habitat patches in the graph. The transformation used to create this gradient from the cost-distances is based on the negative exponential function, considering all potential paths below d_{max} and not just the least cost path. This transformation takes into account the deviation made by the species when passing through any point k not belonging to the least-cost path (see figure 2.3). This gradient is multiplied by a value depending on the parameterization (metric or β). Finally, to obtain a single output raster, all the calculated gradients are overlaid, and at each point, the maximum value is retained (see figure 2.4).

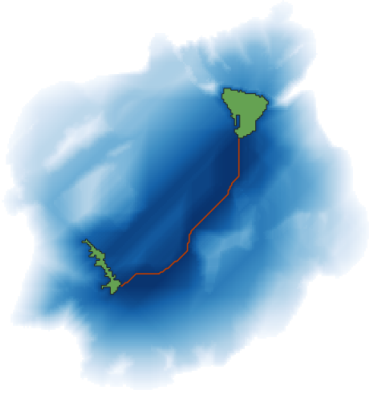


Figure 2.3: Gradient between 2 habitat patches. The least cost path is shown in orange

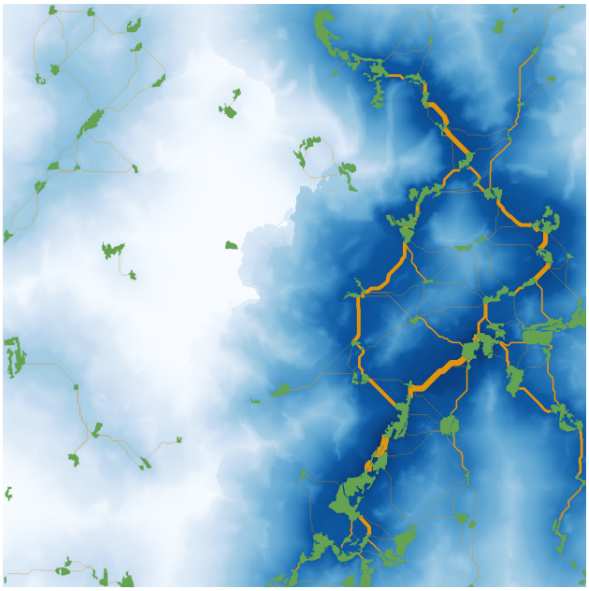


Figure 2.4: Result of a raster corridor calculation parameterized with a metric. Least-cost paths are displayed in orange, with a size proportional to the value of the metric.

Details of the calculation according to the parameter settings are given in the following 2 paragraphs.

Local metric In the case where a local metric is selected the calculation of the gradient G between two patches i and j at any point k is defined such that:

$$Gk_{ij} = M_{ij} e^{-\alpha(d_{ikj} - d_{ij})}$$

With M_{ij} : value of the local metric M on the link between patches i and j, d_{ij} : distance between patches i and j, d_{ikj} : distance between patches i and j passing through point k, α : probability decay parameter as distance increases (cf. Parameters of weighted metrics).

The resulting raster represents a decreasing field of values (cf. figure 2.4). Pixels located on the path of least cost have a value equal to the metric M_{ij}. This value decreases to 0 for a pixel k where $d_{ikj} > d_{max}$.

Parameter β With parameter β the calculation of the gradient G between two patches i and j at any point k is defined as:

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

With a_i: the capacity of patch i, d_{ikj}: distance between patches i and j passing through point k, α: probability decay parameter as distance increases (cf. Parameters of weighted metrics).

The resulting G gradient represents a decreasing field of values. Pixels located on least-cost paths have a value equal to $(a_i a_j)^{\beta} e^{-\alpha d_{ij}}$. This value decreases to 0 for pixel k where $d_{ikj} > d_{max}$.

Result When each Gk_{ij} gradient is calculated, they are overlaid and the maximum value is retained:

$$Ck = \forall i, j \max Gk_{ij}$$

The resulting corridor layer at each point k corresponds to the maximum gradient at that point.

Vector output

In the case of vector output, no other parameters are required.

The result is displayed in a new vector layer, which is automatically saved in the project directory. The layer contains for each link of the selected graph a polygon representing its corridor. Links with a distance greater than d_{max} will have no corridor.

The default view uses transparency to show corridor overlays.

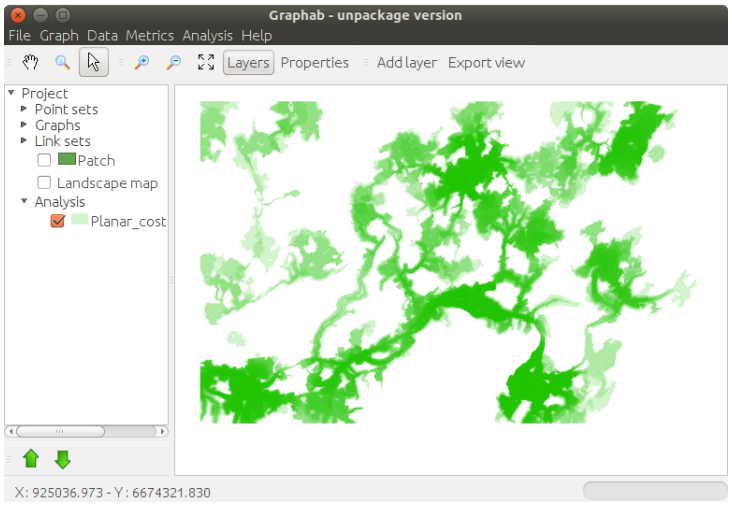


Figure 2.5: Result of a corridor vector calculation

2.4.4 Graph clustering

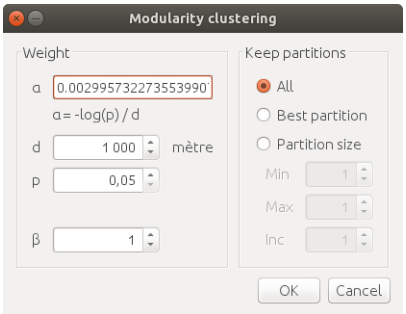
Graphab implements a clustering algorithm that maximises the modularity index [Newman(2006)]. Modularity is a measure of the quality of the clustering of the graph nodes. The underlying principle is that a good clustering involves a large number of links within groups and a small number of inter-group links. The algorithm used is based on the greedy algorithm, followed by a local optimization [Brandes et al.(2008)].

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

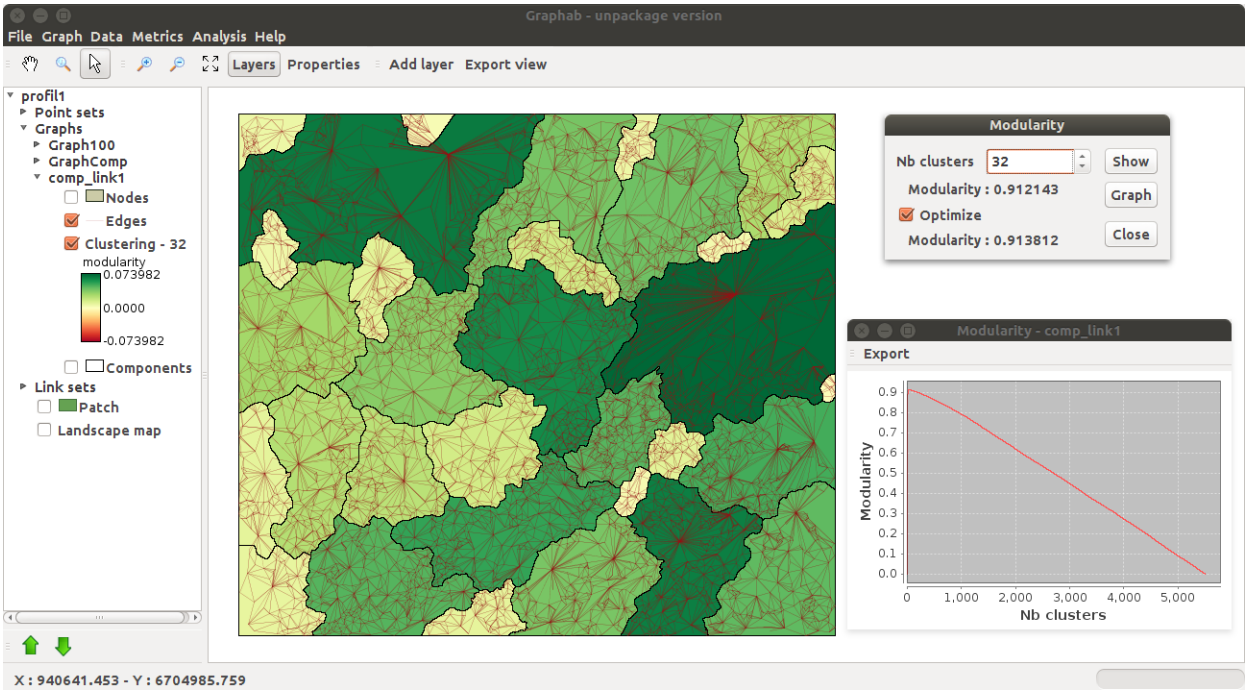
$$w_{ij} = (a_i a_j)^{\beta} e^{-\alpha d_{ij}}$$

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: $w_{ij} = 1$.

The graph clustering is available by right-clicking on the name of the graph. In the first window, the user defines the parameters α et β. The parameter α is not defined directly, it is determined from two other parameters: a distance d and a probability p (cf. Parameters of weighted metrics). To define α to 0, p must be equal to 1.
The right part of the window allows to select which clustering to keep in memory. By default, all clustering are kept in memory to display them after. These settings are useful to avoid memory overflow with large graphs (more than 5,000 nodes). In this case, graphab can keep only the best clustering (ie. with the best modularity) or keep all clustering in a given size interval between 1 and the number of nodes.



After validating this window, the calculation starts until a clustering maximizing modularity is obtained. The result is displayed as a new layer of the graph. Two other windows complete this result.



The new layer shows the elements of the spatial partition as polygons. Since a given element corresponds to a set of patches, it is represented by a polygon resulting from the aggregation of the Voronoï's polygons of all patches of the cluster. The modularity value is used to assign a color to the polygon. The overall modularity is the sum of the modularity values of all clusters.

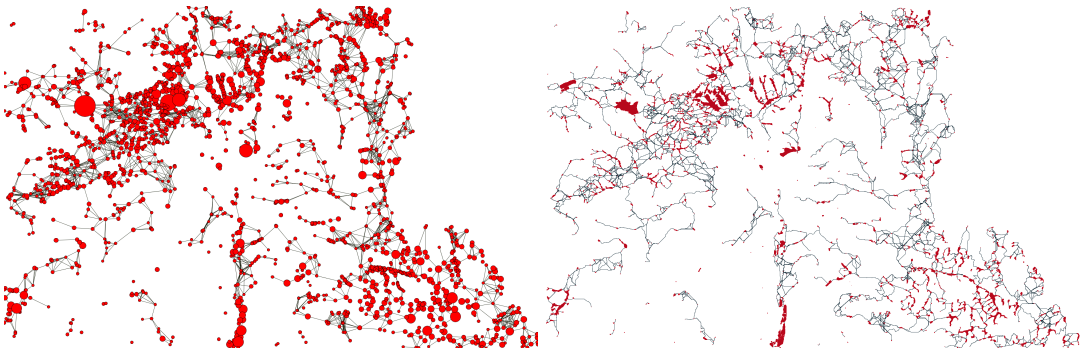
In a first window, a graph shows how modularity varies according to the number of clusters. In the previous figure, the maximal modularity is reached with a small number of clusters.

The second window displays the number of clusters that maximizes modularity and the value of modularity. Several functions are available from this window:
- the number of clusters can be changed to show sub-optimal partitions. The result may be visualized by clicking on the button `Show`. The new layer of partition is added to the graph.
- the button `Graph` allows you to create a new graph including only the intra-cluster links.
- the checkbox `Optimization` achieves a local optimization of the clustering after the use of the greedy algorithm, that significantly improves the modularity value. This option is enabled by default.

2.4.5 Graph visualization

Properties of a graph are available by right-clicking on the name of the graph. Two ways for viewing graphs are available:

- The topologic view displays a simplified view of the graph in which nodes are represented by dots and links by straight lines between centroids.
- The realistic view displays habitat patches according to their actual boundaries and links are represented by least-cost paths between two patches.



Left: Topologic view - Right: Realistic view

2.4.6 Additional functions (context menu)

Certain functions can be accessed via the context menu for each graph in the project tree (right-click on the name of a graph).

Topo view

This entry displays the graph in topological view, see Graph visualization.

Realistic view

This entry displays the graph in realistic view, see Graph visualization.

Clustering

This entry is used to partition the graph using a modularity algorithm, see Graph clustering.

Distance matrix

The `Distance matrix` entry is used to create an array indicating the distance between each pair of nodes for the selected graph. The absence of a connection between two nodes is noted NaN (Not a Number). For more information on its use, see Matrix of inter-patch distances.

Id component

The `Set component id` entry is used to create an attribute in the habitat layer containing the graph component identifier for each patch.

Removing a graph

The `Remove` entry can be used to remove the selected graph. The project is saved automatically. All metrics calculated from this graph are automatically deleted!

Graph properties

The `Properties` entry recalls the parameters used to construct the graph: the graph name, the selected link set, the type of graph with the maximum distance used, if any, and the number of links.

2.5 Calculating connectivity metrics

2.5.1 Metrics family and computing level

Each graph in a project can be used to compute different connectivity metrics. The details of how they are computed and references are listed in the chapter 7. Computations are made at several levels corresponding to major sections in the Metrics menu (table 7.1):

- Metrics /Global metrics: describe the entire graph.
- Metrics /Component metrics: describe connectivity within each component (or sub-graph).
- Metrics /Local metrics: describe the connectivity of each graph element (node or link).
- Metrics /Delta metrics: also describe each graph element, but using a specific computing method. Using the removal method (remove nodes or remove links), the relative importance of each graph element is assessed by computing the rate of variation in the global metric induced by each removal. The result of a delta-metric is at a local level but by reference to the global level.

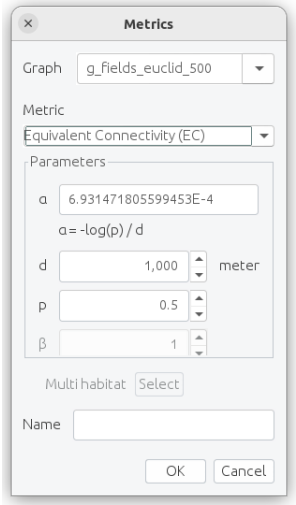
After selecting one of these four computing methods, three families of metrics are available in the new window:

- weighted metrics are based on criteria of distance and patch capacity. They have to adjusted to suit the reference species. These metrics involve computing paths in a graph via Dijkstra's algorithm. After selecting one of these metrics, the user must specify the desired adjustment.
- area metrics are based primarily on the area criterion. If capacity corresponds to a criterion other than patch area, these metrics can be computed and they are expressed in the unit of the criterion used.
- topological metrics are derived from graph theory and they do not require adjustment.

2.5.2 Metrics parameters

Whichever the selected level, the user must first specify the graph on which the calculation will be made and then select the connectivity metric.

Depending on the selected metric, parameters may need to be entered in the `Parameters` zone (see Parameters of weighted metrics). If the graph contains several habitats, you can select/filter the metric calculation according to the desired habitats by checking `Multi habitat` (see Multi-habitat settings). Finally, a name can be given to the result of the metric calculation. If no name is given, a name will be generated automatically, including the short name of the metric, followed by its parameters and the name of the graph.



2.5.3 Parameters of weighted metrics

Alpha parameter

Several metrics include a weighting in their calculation which converts the distance between patches into the probability of movement. These metrics are F, IF, PC, EC, BC... The weighting is based on an exponential function:

$$p = e^{-\alpha d}$$

where p is the probability of movement between two patches, d the distance between these patches, and α a parameter defining the rate of decline in probability as distance increases. As it is not easy to determine the value of the α parameter, Graphab calculates it from the other two parameters. Users must specify the distance corresponding to a certain value of probability, e.g.:

- the maximum dispersal distance of species corresponding to a small value of p (0.05 or 0.01).
- the average dispersal distance of species corresponding to a median value of p (0.5).

The value of α is automatically obtained from the formula:

$$\alpha = -\log(p)/d$$

Beta parameter

The metrics F, IF, CF, and BC are controlled by the β parameter. This parameter is the exponent applied to patch capacity. It adjusts the relative balance between the weight of distances and the weight of patch capacity in the weighting of metrics. Taking the example of the metric F in local computation, whose generic form is:

$$F = \sum a^{\beta} e^{-\alpha d}$$

- a value of $\beta = 0$ means that the patch capacity plays no part in the weighting.
- a value of $\beta = 1$ means that the patch capacity acts linearly in the weighting.
- a value of $\beta = 2$ means that the patch capacity is squared in the weighting.
- a value of $\beta = 0.5$ means that the square root of the patch capacity features in the weighting.
- a value of $\beta = -1$ means that the patch capacity acts in an inversely proportional way in the weighting.

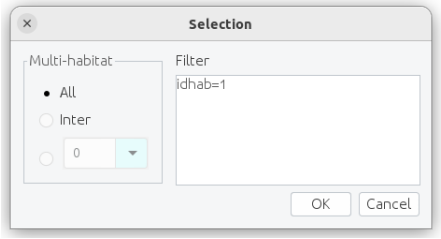
In addition to these few examples, any weighting values are possible.

2.5.4 Multi-habitat settings

For a graph containing several habitats and a metric supporting habitat decomposition (see table 7.1), the checkbox `Multihabitat` can be checked. When checked, the metric result will be decomposed for each habitat.

Selection

The `Selection` button lets you restrict the calculation to certain habitats or graph elements.



The `Filter` part is active only for local or delta metrics, and is used to filter the graph elements for which a result will be calculated.

Filter examples :
`idhab=0`: the metric will only be calculated for nodes in habitat 0
`id=12 || id=20` : the metric will be calculated for nodes 12 and 20 only
`id="1-3"` : the metric will only be calculated for link 1-3
`node` : the metric will only be calculated for nodes in the graph
`edge` : the metric will only be calculated for links in the graph

2.5.5 Display results

Once a metric calculation has been completed, all results are automatically saved in the project. After saving, the metric result is displayed. For a global metric, one (or a few) values will be displayed, for a component metric, the components of the graph will be displayed with a color gradient representing the value of the metric, finally for a local or delta metric, the nodes and links of the graph will be displayed with a color gradient on the nodes or thickness on the links representing the value of the metric.

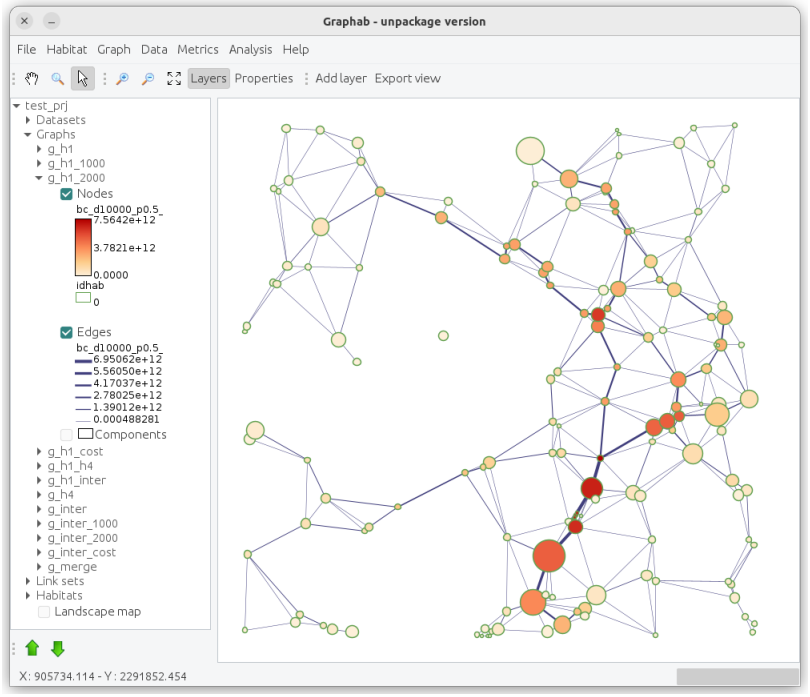
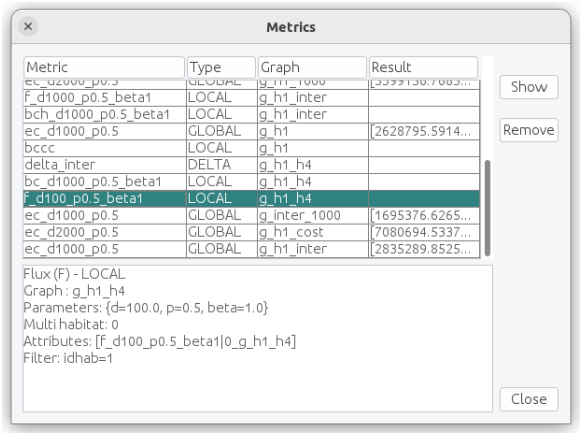


Figure 2.6: Default display of the result of a BC local metric calculation

All calculated metrics can be listed from the menu `Metrics / Show metrics`. In this window, all the parameters of each calculated metric can be found. For global metrics, the result is displayed directly, for other metrics, simply click on the `Display` button to redisplay the metric on the map with the correct graph.



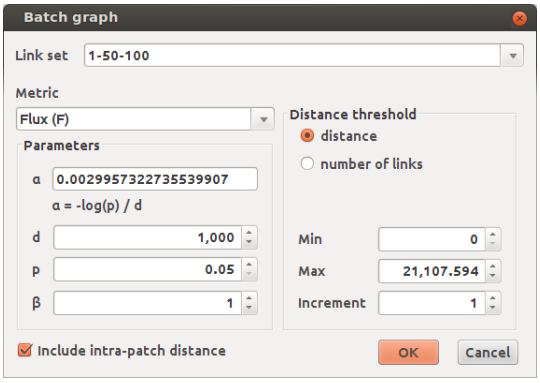
2.5.6 Calculating batch metrics

Every metric compatible with the global level can be calculated following the variation of the scale of distances. This variation may concern either graph pruning (2.5.6), or metric adjustment (2.5.6). The type of distance used depends on the type of distance used in creating the link set (Euclidean, least-cost distance, or least-cost path).

Batch graph

The Metrics / Batch graph menu allows a series of pruned graphs to be created from a given link set and a metric to be calculated for each graph at global level. The maximum distance of successive graphs are increasingly defined in either of two ways:

- distance: a fixed increment of distance is defined between successive graphs. The metric values are therefore calculated inregular intervals of distance.
- number of links: a fixed number of links is defined between successive graphs. This number of links is automatically converted into distance used to prune the graph. These distances may be unevenly spread.



Graphs are defined following three criteria selected by users:

- min: smallest distance used for the first graph in the series. By default, this minimum is 0, corresponding to the total absence of links.
- max: maximum distance used for the final graph in the series. By default, this maximum corresponds to the maximum distance or to the link number of the selected link set.

- increment: distance value added between each new graph.

Once the calculation is completed, the software opens a new window displaying the curve of the selected metric versus the maximum distance. The values of this curve can be saved with the "Export" button by selecting text format.

Batch parameter

The Metrics / Batch parameter menu is used to calculate a series of metrics from a given graph. This procedure applies to the weighted metrics only. It is divided into two entries: local metrics or global metrics.

Batch parameter for local metrics

A local weighted metric is calculated in series according to the variation of one of its parameters. The user must select the graph, the metric, and the parameter to be varied. The variation of computation is defined by:

- min: minimum value of the parameter,
- max: maximum value of the parameter,
- increment: interval value between two metric computations.

Once the calculation is completed, the patches (and in some cases the links) of the graph are characterized by a series of additional metrics.

Batch parameter for global metrics

For a given graph, a global weighted metric is calculated in series according to the variation of one of its parameters. As previously, this variation is defined between a minimum value (min), a maximum value (max), and with an interval (increment).

The procedure ends with the opening of a new window displaying the curve of the selected metric versus the parameter.

Table 2.1 summarizes possible metrics calculations.

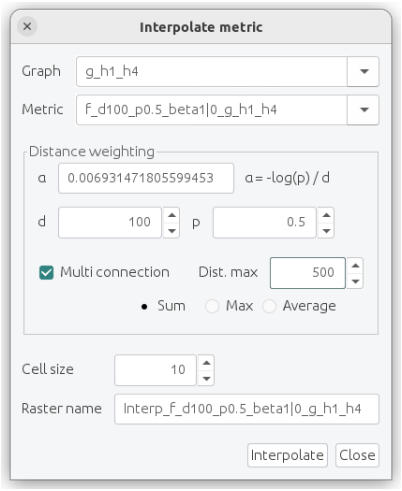
Family	Connectivity metrics	Code	Patch capacity	Intra-patch distance	Parameters		Batch graph	Batch param
					α	β		
Weighted	Flux	F	×	×	×	×	×	×
	Equivalent Connectivity	EC	×	×	×		×	×
	Probability of connectivity	PC	×	×	×		×	×
	Interaction Flux	IF	×	×	×	×		×
	Fractions of delta Probability of connectivity	dPC	×	×	×			
	Betweenness centrality index	BC	×	×	×	×		×
	Integral index of connectivity	IIC	×				×	
	Current flow	CF	×			×		×
Area	Mean size of the components	MSC	×				×	
	Size of the largest component	SLC	×				×	
	Class coincidence probability	CCP	×				×	
	Expected cluster size	ECS	×				×	
Topological	Node Degree	Dg						
	Clustering coefficient	CC						
	Closeness centrality	CCe		x				
	Eccentricity	Ec		x				
	Connectivity correlation	CCor						
	Number of components	NC					×	
	Graph diameter	GD		×			×	
	Harary Index	H					×	
	Wilks' Lambda	W					×	

Table 2.1: Possible connectivity metrics calculations

2.6 Analysis

2.6.1 Interpolating metrics

The `Analysis / Metric interpolation` menu is used to create raster layers from local metrics calculated at patch level. This transformation is based on a specific spatial interpolation which assigns connectivity values of patches to each cell of a grid, using a decreasing weighting function from the patch edge (weight of 1). Overall, the farther cells are away from the graph, the lower their connectivity values.



The weighting is a negative exponential function as $p = e^{-\alpha d}$ for which the user selects a distance (d) corresponding to a certain probability (p) and the software deduces the value of the α parameter. In theory, this adjustment must be consistent with the choice of reference graph or of any weighted metrics, using the same value of d.

The `Multi connection` option allows several patches to be included in the calculation of metrics at the point level. The calculation is based on a sum or a weighted mean of values of all patches in the vicinity of the points, up to the specified Maximum distance.

The distance used in these calculations depends on the reference graph. If it is based on least-cost distance, the spatial interpolation uses the same distance and not Euclidean distance.

2.6.2 Patch addition

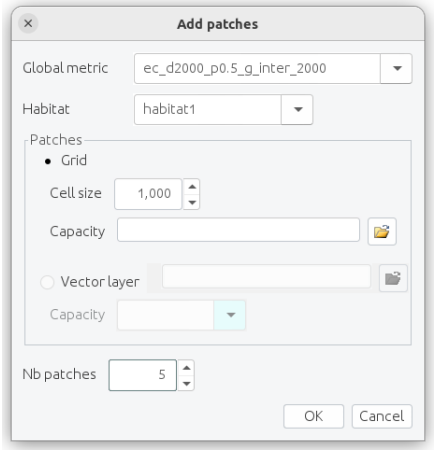
The function `Add patches` is available in the menu `Analysis`. It is used to test a set of locations to create new habitat patches according to the connectivity gain they provide. Graphab adds virtually a patch and the links between this patch and existing patches (only if their distance is less than the maximum distance of the graph in the case of a pruned graph) and calculates the global metric again. Once all locations are tested, the software validates the one for which patch addition produces the biggest increase in the metric value. The process is repeated until the desired number of additional patches is reached by including elements already added at each step.

Several parameters must be defined:

- Global metric: the global metric to maximize. The metric must have been calculated before on a graph which must not include intra-patch distance and the underlying link set must be in complete topology.
- Habitat: if the metric has been calculated on several habitats, select the habitat that will contain the new patches.
- Nb Patches: number of patches (which become new nodes of the graph) to be added.

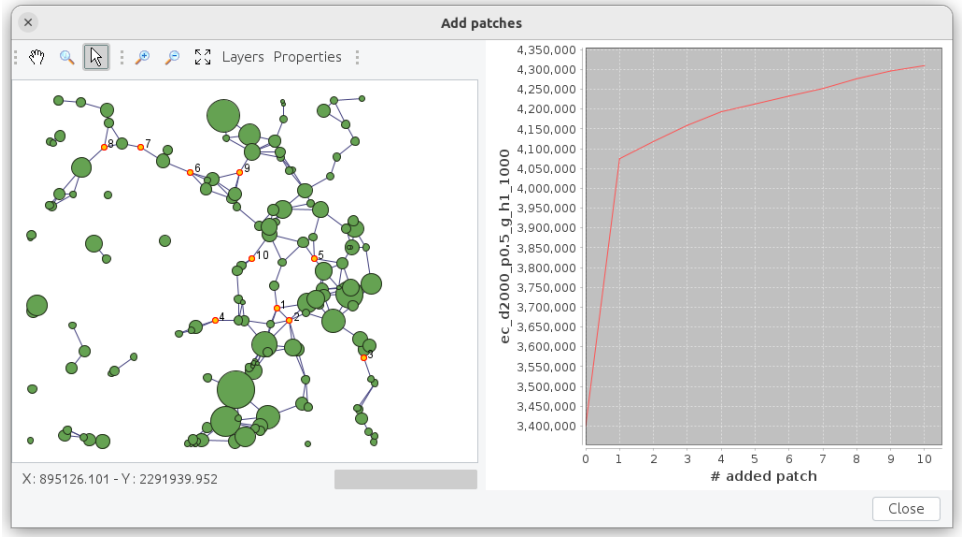
The definition of the patch set to test can be done in two ways:

- by automatically applying a grid of a given resolution (field "cell size" to define). In this case, Graphab tests each centroid of the cells by adding a virtual new patch. By default, all centroids have the same capacity value (=1). Nevertheless, it is possible to import a raster layer in which each pixel of the lansdcape matrix has its own capacity value,
- by importing a shapefile of points or polygons. In this case, the user must indicate which column of the attribute table contains the patch capacity.



At the end of the calculation, a new window appears with 2 parts:

- on the left is the graph with the added patches numbered from 1 to n.
- on the right the evolution of the global metric after the addition of each new patch.



All the results are automatically saved in a folder named `addpatch_nX-GraphX_...` in the project directory. It contains:

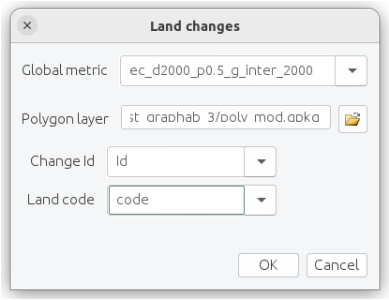
- a layer containing the added nodes. The attribute table indicates the step at which each node was validated and the corresponding value of the global metric,
- two layers containing all the links of the graph, including those connecting the new nodes in topological form `topo_links_graphXXX.gpkg` or in realistic form `links_graphXX.gpkg`,
- a folder named `detail` containing the layer of the tested locations and the corresponding value of the global metric for each step.

References : [Clauzel et al.(2015a), Foltête et al.(2014)]

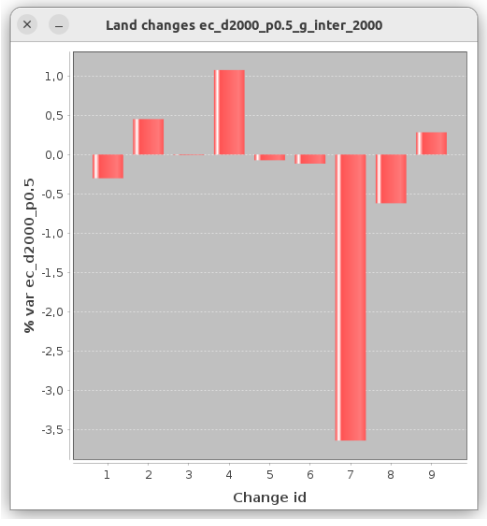
2.6.3 Land changes

Once graphs and metrics have been calculated in Graphab, it may be useful to test land use changes and measure their impact on the connectivity of the ecological network being modeled. Graphab already contains several specific graph modification functions (delta metric, Patch addition and several CLI commands Adding/Removal). However, these functions only manage the modification of the graph by adding or deleting elements. To test more detailed scenarios, it may be more interesting to go back to the landscape map itself to precisely define the changes in land use. These changes can impact the elements of the graph in different ways: creation/deletion of elements, but also the capacity of nodes or the distances of links.

From the `Analysis/Land changes` menu, you can test a set of changes to the landscape map and measure their impact on a global connectivity metric.



Select a global metric already calculated on a graph, and a layer of polygons containing the modifications to be applied. Each modification to be tested corresponds to one or more polygons with the same identifier and a land use code specific to each polygon, which will be replaced on the map at the polygon's location. From this modified map, a new project will be created with all the elements needed to recalculate the selected global metric (habitats, link sets and graphs) in a sub-directory of the project named by the identifier. This procedure will be iterated for each polygon layer identifier. As a result, a graph is displayed showing the evolution of the global metric for each polygon layer identifier. A `landmod_metric_name.csv` file is automatically saved in the project folder, containing the global metric value for each identifier.



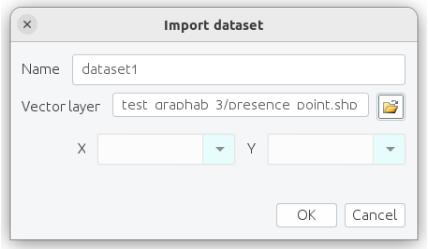
As this method recreates a complete project for each modification test, processing can be very cumbersome and heavier than methods for adding/removing elements on the graph.

2.7 External datasets

The main part of the software is for graph construction and computation of connectivity metrics. But it is often useful to connect these elements with external data. Graphab allows graph data to interact with external datasets, or even integrate this dataset directly into a graph.

2.7.1 Importing datasets

Datasets can be imported via the `Data / Import dataset` menu. These data must be correspond to a vector layer, often representing point data.



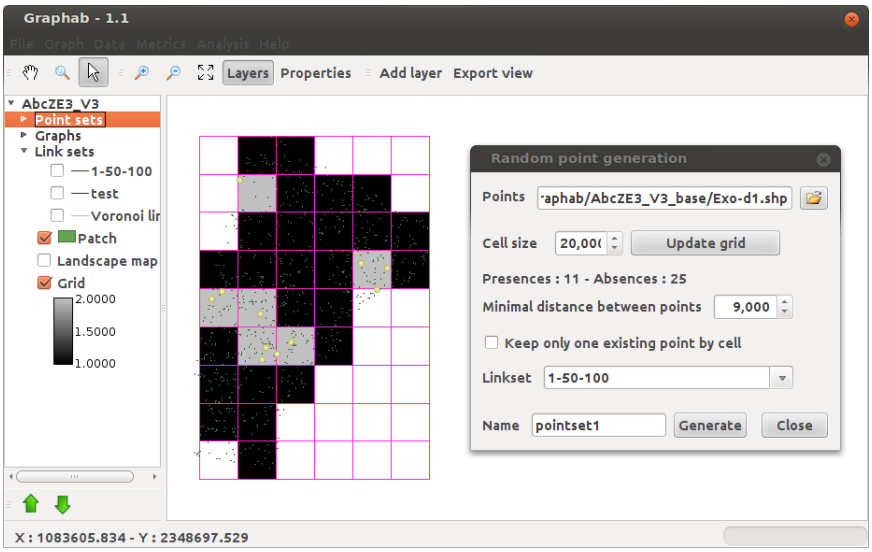
The imported file may be either a vector layer in geopackage (*.gpkg) or shapefile (*.shp) format, or in table format (*.csv). For files in table format, the user must specify the columns corresponding to the XY coordinates of points in the table.

In the project, the new dataset appears in the tree under `Datasets`. It corresponds to a particular habitat with zero patch capacity. This abstraction is useful to create link sets and graphs, and to calculate metrics on these external datasets.

If the data represent points of presence of a species with no points of absence, the user can create a pseudo-absence dataset cf. Generating random points.

2.7.2 Generating random points

The `Data / Generate random points` menu can be used to generate a set of pseudo-absence points based on a set of presence points.



The user must load a file of presence points and specify the name of the set of presence/pseudo-absence points to be created.

Several parameters must be defined to randomly sample absence points:

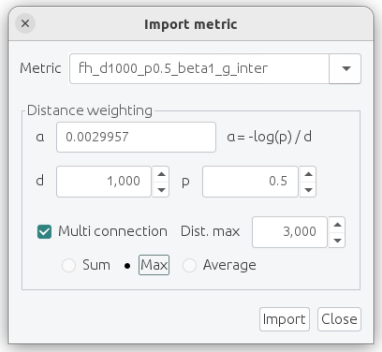
- Cell size of the grid (in meters) to define the size of cells from which absence points will be potentially sampled. The "update grid" button can be used to display the grid according to the selected cell size.
- Minimum distance between points: this function reduces the effects of spatial autocorrelation by specifying a minimum distance in meters to be observed between the generated absence points and between these points and presence points.
- Type of distance: the unit of the minimum distance between points depends on the type of the distance used in the link set selected.
- Keep only one existing point by cell: this option (checked box) retains only one presence point in each cell thereby reducing the effects of spatial autocorrelation.

2.7.3 Additional functions (context menu)

Certain functions can be accessed via the context menu for each dataset in the project tree (right-click on the name of a dataset).

Import local metric

This function brings back a local metric calculated on the nodes of a graph to the elements of the dataset. The calculation uses the same principle as metric interpolation (see section 2.6.1). But instead of interpolating a metric at any point in space, the metric will be interpolated only for the elements in the dataset, which will be much faster.



Distance matrix

The `Distance matrix` entry creates a table showing the distance between each pair of elements in the selected dataset. For more information see: Matrix of inter-patch distances.

Removing a dataset

The `Remove` entry deletes the selected dataset. After deletion, the project is automatically saved. All link set, graphs and metrics calculated from this dataset are automatically deleted!

2.8 Display

2.8.1 Context menu

Properties

The properties of link sets, graph elements (nodes, links, and components), and point data are available by right-clicking on each of them.

Style

The Style menu includes the display parameters for objects: color, line width, label, symbol size (for nodes only). Objects can be represented in the same way (single symbol) or according to some attribute. A discretization method can be applied to classify objects according to the values of the selected attribute. By default, the legend of objects is displayed in the table of contents. It can be masked by unchecking the Legend button.

Export

The Export menu can be used to export objects to a shapefile (*.shp), a geopackage (.gpkg) or a text file (*.txt).

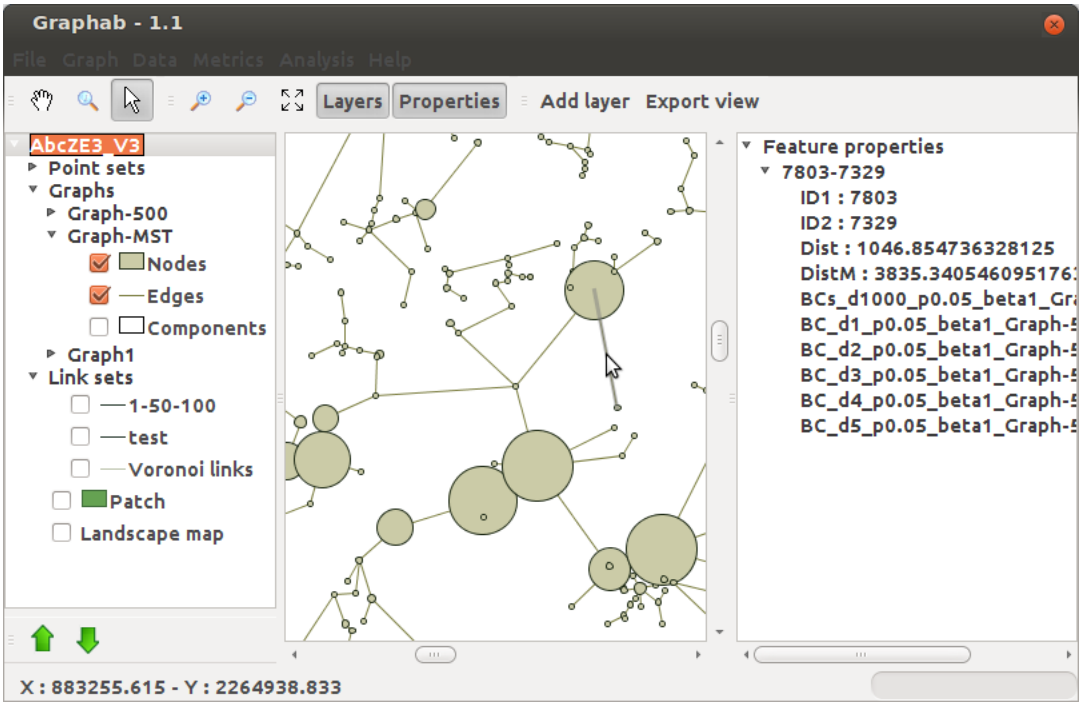
Statistics

The `Statistics` menu displays the distribution of the values of one or more attributes:

- scatter plot: values of two attributes are plotted on a two-dimensional graph,
- histogram: the bar chart of the values of an attribute is generated.

2.8.2 Properties of map elements

It is also possible to display the values of a given object by selecting it with the white arrow. After selection, the values of attributes are displayed in a new right-hand column named “Feature properties”. This column can be closed by clicking on the Properties menu in the top bar.


Graphab - 1.1
File Graph Data Metrics Analysis Help
Layers
Properties
Add layer
Export view
AbcZE3_V3
Point sets
Graphs
Graph-500
Graph-MST
Nodes
Edges
Components
Graph1
Link sets
1-50-100
test
Voronoi links
Patch
Landscape map
Feature properties
7803-7329
ID1 : 7803
ID2 : 7329
Dist : 1046.854736328125
DistM : 3835.340546095176
BCs_d1000_p0.05_beta1_Gra
BC_d1_p0.05_beta1_Graph-5
BC_d2_p0.05_beta1_Graph-5
BC_d3_p0.05_beta1_Graph-5
BC_d4_p0.05_beta1_Graph-5
BC_d5_p0.05_beta1_Graph-5
X : 883255.615 - Y : 2264938.833

Chapter 3

Command line interface

3.1 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-3.0.jar --help
```

Result:

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

You're ready to use Graphab in CLI mode.

3.2 Syntax

3.2.1 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, ...`

3.2.2 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.

3.2.3 Optional parameter

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

3.2.4 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.

3.2.5 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-3.0.jar --project prj.xml --graph --gmetric NC
```

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

Chapter 4

Command reference

4.1 General command

4.1.1 –help : display help

Command :

```
java -jar graphab-3.0.jar --help
```

Result :

```
Usage :
java -jar graphab.jar --metrics
java -jar graphab.jar [-proc n] --create prjname landrasterfile [nodata=val] [dir=path]
java -jar graphab.jar [-mpi | -proc n] [-nosave] [-distconv excost=val] --project prjfile.xml command1
Commands list :
--show
--show_habitats --show_linksets --show_graphs --show_metrics
--dem rasterfile
--habitat name=habitatname codes=code1,...,coden [minarea=val] [maxsize=val] [con8] [novoronoi]
--vhabitat name=habitatname file=vectorlayer|from=habitatname [mincapa=val] [capa=field] [novoronoi]
--usehabitat habitat1,...,habitatn
--removehabitat [habitat1,...,habitatn]
--mergehabitat [habitat1,...,habitatn] [novoronoi]
--capa [area [exp=value] [code1,..,coden=weight ...]] | [file=capacity.csv id=fieldname capa=...
--linkset distance=euclid|cost [name=linkname] [topo=planar|complete|planarcost] [inter] [maxcost=...
--uselinkset linkset1,...,linksetn
--removelinkset [linkset1,...,linksetn]
--graph [name=graphname] [nointra] [threshold=[{]min:inc:max[}]]
--usegraph graph1,...,graphn
--removegraph [graph1,...,graphn]
--mergegraph [name=graphname] [graph1,...,graphn]
--cluster d=val p=val [beta=val] [nb=val]
--metapatch [mincapa=value] [novoronoi]
--corridor maxcost=[{]min:inc:max[}] [format=raster|vector] [beta=exp|var=name d=val p=val]
--dataset name=dataname file=vectorlayer.gpkg
--removedataset [dataset1,...,datasetn]
--distance_matrix type=space|graph distance=euclidean|leastcost|circuit|flow|circuitflow [d=val p=val]
--gmetric global_metric_name [resfile=file.txt] [mh=all|inter|val] [param1=[{]min:inc:max[}] ...
--cmetric global_metric_name [mh=all|inter|val] [param1=[{]min:inc:max[}] [param2=...
```

```
--lmetric local_metric_name [mh=all|inter|val] [filter=condition] [param1=[{]min:inc:max[}] ...
--usemetric metric1,...,metricn
--removemetric [metric1,...,metricn]
--interp [var=patch_var_name d=val p=val] [name=rastername] [resol=val] [multi=dist_max ...
--model variable distW=[{]min:inc:max[}] [vars=var1,...,varn] [raster=r1,...,rn]
--delta [filter=condition | [obj=patch|link [sel=id1,id2,...,idn | fsel=file.txt]]
--addpatch npatch=val [hab=name] gridres=min:inc:max [capa=capa_file] | patchfile=file.gpkg ...
--remelem nstep global_metric_name [param1=val ...] obj=patch|link [sel=id1,id2,...,idn|fsel=file.txt]
--gtest nstep global_metric_name [param1=val ...] obj=patch|link sel=id1,id2,...,idn|fsel=file.txt
--gremove global_metric_name [param1=val ...] [patch=id1,id2,...,idn|fpatch=file.txt] [link=...
--landmod zone=filezones.gpkg id=fieldname code=fieldname [sel=id1,id2,...,idn]
--landmodgm zone=filezones.gpkg id=fieldname code=fieldname [sel=id1,id2,...,idn]

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

4.1.2 –metrics : list available metrics

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

Command :

```
java -jar graphab-3.0.jar --metrics
```

Result :

```
===== Global metrics =====
S#F - Sum Flux (F)
params : [d, p, beta]
EC - Equivalent Connectivity  (EC)
params : [d, p]
PC - Probability of Connectivity (PC)
params : [d, p]
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]
W - Wilks index (W)
params : [attrs, npatch, warea]

===== Local metrics =====
F : Flux (F)
params : [d, p, beta]
BC : Betweeness centrality (BC)
params : [d, p, beta]
IF : Interaction Flow (IF)
params : [d, p, beta]
CF : Current flow (CF)
params : [beta]
Dg : Node degree (Dg)
```

```
CC : Clustering Coefficient (CC)
CCe : Closeness centrality (CCe)
CCor : Connectivity Correlation (CCor)
Ec : Eccentricity (Ec)
```

4.2 Project management

4.2.1 –create : create project

`java -jar graphab-3.0.jar --create` *`prjname landrasterfile`* `[nodata=`*`val`*`] [dir=`*`path`*`]`

Required parameters

- `prjname`: name of the new project
- `landrasterfile`: filename containing land cover in tiff, ascii grid or rst format

Optional parameters

- `nodata=val`: code for nodata
- `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.

4.2.2 –project : load project

This command sets the path to the project xml file.

`java -jar graphab-3.0.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.

4.2.3 –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-3.0.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.

4.2.4 –dem : import DEM

This command imports a DEM in the project for calculating slope in linkset creation.

```
--dem rasterfile
```

Mandatory parameter

- `rasterfile` : raster file in tiff or ascii grid format. The grid geometry must be the same as the landscape map.

4.3 Habitat

4.3.1 –habitat : creation of a habitat from the landscape map

```
--habitat name=habitatname codes=code1,...,codeN [minarea=val]
[maxsize=val] [con8] [novoronoi]
```

Required parameters

- `habitatname` : name of the new habitat
- `codes=code1,...,coden` : the land use code(s) representing the habitat patches.

Optional parameters

- `minarea=val` : minimum size of a habitat patch in hectares.
- `maxsize=val`: cut out patches whose length or width in meters exceeds `maxsize` on a size grid `maxsize`
- `con8`: connectivity to 8 neighboring pixels for the definition of spots, by default the connectivity is only 4
- `novoronoi`: does not calculate the voronoi diagram for planar topology. This option is useful for saving time if planar topology is not used when creating the link set.

Description

This command creates a new habitat from the landscape map

The following commands will be executed on the newly created habitat.

4.3.2 –vhabitat : create a habitat from a vector layer

```
--vhabitat name=habitatname file=vectorlayer|from=habitatname [mincapa=val]
[capa=field] [novoronoi]
```

Required parameters

- `habitatname` : name of the new habitat
- `file=vectorlayer|from=habitatname` : the data source for this habitat. From a vector layer in .gpkg or .shp format (`file`) or from an existing habitat in the project (`from`). This second option is useful for filtering out patches with too low capacity with the `mincapa` parameter.

optional parameters

- `mincapa=val` : minimum capacity of a patch to be kept
- `capa=field` : the name of the vector layer field containing the capacity. Useful only for the `file` parameter.
- `novoronoi`: does not calculate the voronoi diagram for planar topology. This option is useful for saving time if the planar topology is not used when creating the linkset.

Description

This command creates a new habitat from a vector layer independently of the landscape map. The extent of this layer must be within the extent of the landscape map. This command can also be used to create a habitat from an existing one, to filter out patches with too low capacity, using the `from` parameter in conjunction with the `mincapa` parameter.

The following commands will be executed on the newly created habitat.

4.3.3 –usehabitat : habitat selection

`--usehabitat ALL|textithabitat1,..,habitatn`

Selects the habitats to be used in subsequent commands. Habitat names are case-sensitive and cannot contain spaces. The keyword `ALL` can be used to select all habitats.

By default, all habitats in a project are selected.

4.3.4 –removehabitat : habitat removing

`--removehabitat ALL|textithabitat1,..,habitatn`

Removes selected habitats. All link sets, graphs and metrics dependent on these habitats will also be deleted. The `ALL` keyword can be used to delete all habitats.

The `-nosave` option has no effect on this command.

4.3.5 –mergehabitat : merge habitats

`--mergehabitat [ALL|`*habitat1,..,habitatn*`] [novoronoi]`

Optional parameters

- |verb|habitat1,..,habitatn| : list of habitats to be merged. If this list is not given, the currently selected habitats will be merged.
- `novoronoi`: does not calculate the voronoi diagram for planar topology. This option is useful for saving time if the planar topology is not used when creating the link set.

Merges selected habitats into a single habitat. This command is useful for calculating a link set connecting all these habitats.

The following commands will be executed on the merged habitat.

4.3.6 –capa : set patch capacity

```
--capa [area [exp=value] [code1,..,coden=weight ...]]
| [file=capacity.csv id=fieldname capa=fieldname]
| [maxcost=[{]valcost[}] codes=code1,code2,...,coden [weight]]
```

Area parameters

- `area`: set capacity to the patch area (default)
- `exp=value`: apply an exponent to the area.
- `code1,..,coden=weight ...`: weights of land cover codes used for defining patches. This parameter is useful only when patches are defined with several land cover codes.

External file parameters

- `file=capacity.csv`: defines capacity from given csv file
- `id=fieldname`: name of column in file containing patch identifier
- `capa=fieldname`: name of the file column containing the new capacity value.

Neighborhood 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 or with `area` parameter, the capacity is defined as the area of the patch in m^2.
With the `file=` parameter, capacity is defined from an external file in csv format.
With `maxcost` parameter, 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 the resolution for all land cover codes.

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

Examples

```
--capa
```

```
--capa area
```

The two above examples calculates the capacity as the patch area.

```
--capa area 1=0.5 5=2
```

Set the patch capacity to the patch area (in m^2) weighted by 0.5 for land cover code 1 and 2 for land cover 5. The patches have to be defined by land cover codes 1 and 5.

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

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

4.3.7 –metapatch : create meta-patch habitat

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

Optional parameter

- `mincapa=value`: minimum capacity of a meta-patch to be retained in the new habitat
- `novoronoi`: does not calculate the voronoi diagram for planar topology. This option is useful for saving time if the planar topology is not used when creating the linkset.

Description

The `--metapatch` command creates a new habitat with meta-patch based on each 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.

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

Example

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

This command creates a meta-patch habitat 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)]

4.4 Link set

4.4.1 –linkset : create linkset

`--linkset distance=euclid|cost [name=`*linkname*`] [topo=planar|complete|planarcost] [inter] [maxcost=`*valcost*`] [slope=`*coef*`] [remcrosspath|nopathsaved] [[`*code1,..,coden=cost1 ...] codei,..,codej=min:inc:max* `| extcost=`*rasterfile*`]`

Mandatory parameter

- `distance=euclid|cost`: set euclidean distance or raster based distance (cost). For cost distance, costs must be sets (see below).

Optional parameters

- `name=linkname`: link set name to create
- `topo=planar|complete` : set the linkset topology. Default topology is planar (`planar`)
- `inter` : creates a set of inter-habitat links ie. retains only links between different habitats
- `maxcost=valcost`: limit path calculation up to the given distance *valcost*.

Parameters for the case distance=cost

- `topo=planarcost`: this option lets you determine planar topology using cost distances instead of euclidean distances.
- `slope=coef`: weight costs with the slope. For using this option, the project must contain a DEM.
- `remcrosspath`: remove links crossing patches
- `nopathsaved`: do not save the paths geometry, useful for reducing memory consumption with complete topology
- `[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 topo=complete 1,2,3,4,5,6,7=1
```

You can prescribe a threshold to avoid the creation of too many links (threshold = 100):

```
--linkset distance=cost topo=complete maxcost=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.

4.4.2 –uselinkset : select linkset

```
--uselinkset ALL|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.

4.4.3 –removelinkset : remove linkset

`--removelinkset ALL|[`*`linkset1,..,linksetn`*`]`

Remove selected linksets from the project. All graphs and metrics depending on removed linksets are also removed.
Global option `-nosave` has no effect on this command.

4.5 Graph

4.5.1 –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 unpruned graph 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 pruned 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.

4.5.2 –usegraph : select graph

`--usegraph ALL|`*`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.

4.5.3 –removegraph : remove graph

`--removegraph ALL|[graph1,..,graphn]`

Remove selected graphs from the project. All metrics calculated on these graphs are also removed.
Global option `-nosave` has no effect on this command.

4.5.4 –mergegraph : merge graphs

`--mergegraph [name=graphname] ALL|[graph1,..,graphn]`

Optional parameters

- `name=graphname` : name of the graph to be created.
- `ALL|graph1,..,graphn` : list of graphs to be merged or the keyword `ALL` to merge all graphs.

Merges the selected graphs into a single graph. Selection can be made in 2 ways: either by a selection preceding this command, or by a list of graphs given to this command. The following commands will be executed on the newly created graph.

Examples

Without parameters, the command will create a graph merging the previously selected graphs

`--mergegraph`

The command below creates a graph named gm, merging graphs g1 and g2

`--mergegraph name=gm g1,g2`

4.5.5 –corridor : calculate corridors

`--corridor maxcost=[{]min:inc:max[}] [format=raster|vector] [beta=exp|var=name d=[{]val[}] p=v`

Required parameter

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

Optional parameter

- `format=raster|vector` : output format raster (.tif) or vector (.gpkg). The default format is vector.

Optional parameters for raster format

If the parameter `beta` or `var` is set, the pixel no longer corresponds to the number of corridors but to the maximum value from the corridors. At the level of each corridor a decreasing gradient is calculated from the shortest paths. This gradient takes the value 1 at the shortest paths and decreases following the inverse exponential law: $e^{-\alpha d}$.The result of this gradient is multiplied by the product of the patch capacities $(a_i a_j)^{\beta}$ if the parameter `beta` is set or by a local metric if the parameter `var` is defined. The parameters `beta` et `var` are exclusive and the parameters `d` et `p` must be set to define the α exponent of the distance decay.

- `beta=exp` : exponent of the product of the capacities of the patches $(a_i a_j)^\beta$. The usual values are 0 : no effect of the capacities, 0.5 : geometric mean of the capacities or 1 : product of the capacities.
- `var=name` : `name` corresponds to a local metric computed on the links
- `d=[{]val[}]` : distance to define the exponent α of distance decay
- `p=val` : probability to define the exponent α of distance decay

Description

The `--corridor` command calculates the corridor associated to each link of a graph. The result is stored in a shapefile (or raster file) for each selected graph. For each link, the shapefile contains a polygon representing the set of paths having a distance less than or equal to *valcost*. In raster format, by default, each pixel counts the number of corridors passing through it. If the parameter `beta` or `var` is set, the calculation is different (cf. Raster output).

Example

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

4.5.6 –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 defaut 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: $w_{ij} = 1$. For more details, see Graph clustering.

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

Reference

[Foltête and Vuidel(2017)]

4.6 Calculate metric

4.6.1 –gmetric : calculate global metric

`--gmetric` *global_metric_name* [`resfile=`*file.txt*] [`mh=all|inter|`*val*] [*param1=min:inc:max* [*param2=min:inc:max* ...]]

Required parameter

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

Optional parameters

- `resfile=file.txt` : file containing the resulting values of the metric
- `mh=all|inter|val`: decomposition of the metric according to habitats: `all` creates as many results as possible habitat combinations, `inter` creates a single result between different habitats, `val` specifies the desired habitat combination (ex: `0/0`)
- `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. Another file name can be given by the `resfile` parameter.

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
```

4.6.2 –cmetric : calculate component metric

`--cmetric` *`global_metric_name`* `[mh=all|inter|`*`val`*`]`
`[`*`param1=min:inc:max`* `[`*`param2=min:inc:max`* `...]]`

Required parameter

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

Optional parameters

- `mh=all|inter|val`: decomposition of the metric according to habitats: `all` creates as many results as possible habitat combinations, `inter` creates a single result between different habitats, `val` specifies the desired habitat combination (ex: `0/0`).
- `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.

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

4.6.3 –lmetric : calculate local metric

`--lmetric` *`local_metric_name`* `[name=`*`resname`*`] [mh=all|inter|`*`val`*`] [filter=condition]` `[`*`param1=min:inc:max`* `[`*`param2=min:inc:max`* `...]]`

Required parameter

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

Optional parameters

- `name=resname`: set the metric result name
- `mh=all|inter|val`: decomposition of the metric according to habitats: `all` creates as many results as possible habitat combinations, `inter` creates a single result between different habitats, `val` specifies the desired habitat combination (ex: `0/0`).
- `filter=condition` : determines for which elements of the graph a calculation will be performed and stored cf. Parameter filter=condition.
- `param1=[{]min:inc:max[}]` : metric parameter(s), if needed. Surrounded by braces, the distance values are converted automatically in cost values.
- ...

Description

Calculates the given local metric on each element (node and/or link) of each selected graph. The name of the metric is its short name as displayed by the `--metrics` command. The result is stored in an attribute of the patches and/or links. The result is saved in the project unless `-nosave` option is used.

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

4.6.4 Metrics parameters

Parameter filter=condition

The `--lmetric` and `--delta` commands have a parameter for selecting the elements on which the calculation will be performed.

Apply the calculation to graph nodes only:

```
filter=node
```

Apply the calculation only to nodes in the graph whose habitat identifier is 1:

```
"filter=node && idhab=1"
```

Note that if you include spaces in the condition, you must enclose the set with double odds.

Apply the calculation only to graph nodes with a surface area greater than $1000m^2$:

```
"filter=node && area > 1000"
```

Apply the calculation only to links in the graph:

```
filter=edge
```

If the metric contains the parameters `d` and `p` then 2 other optional parameters can be used: `maxd` or `minp`.

Parameter maxd=valcost

The `maxd` parameter limits path calculation to *valcost* (i.e. paths greater than *valcost* will not be calculated). This parameter can significantly reduce the calculation time of a metric based on path calculation, but the result will be less accurate.

Parameter minp=valproba

The `minp` parameter limits path calculation to a probability greater than *valproba*. This parameter can significantly reduce the computation time of a path-based metric, but the result will be less accurate. This parameter is equivalent to `maxd`, except that it applies to probability instead of distance. This parameter is useful when varying the distance parameter.

4.6.5 –interp : interpolate a metric

`--interp [var=`*patch_var_name* `d=`*val* `p=`*val*`] [name=`*rastername*`] [resol=`*val*`] [multi[=`*dist_max*`] [ag=sum|max|avg]]`

Optional parameters

- `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. The metric short name can be used.
- `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
- `name=rastername` : name of the raster file storing the result of the interpolation
- `resol=val` : resolution of the interpolation. In some cases, this parameter is ignored and the resolution is set to the land cover map resolution
- `multi[=dist_max]` : interpolation from all patches within the *dist_ max* distance, instead of the nearest patch only
- `ag=sum|max|avg` : aggregation by a sum, maximum or a weighted average. Used only with the `multi` parameter. Sum is the default aggregation.

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 :

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

Examples

The `--interp` command can be used without parameters if one or more local metrics are calculated or selected previously.

```
--lmetric F d=1000 p=0.5 beta=1 --interp multi=5000 ag=sum
```

The α parameter normally defined by the `d` and `p` parameters is defined using the metric parameterization. Same example, giving the metric name :

```
--interp var=f_d1000_p0.5_beta1_2000m_euclid d=1000 p=0.5 multi=5000 ag=sum
```

Same command as above, only the attribute name has been reduced for simplicity. In this case all calculated F metrics will be interpolated.

```
--interp var=f_ d=1000 p=0.5 multi=5000 ag=sum
```

4.7 Dataset and SDM

A dataset is a vector habitat whose capacities are zero. This abstraction makes it possible to compute link sets, graphs and metrics that include a dataset in addition to the usual habitats. As a result, in most commands a habitat can be replaced by a dataset.

4.7.1 –dataset : import dataset

`--dataset name=`*`dataname`* `file=`*`vectorlayer.gpkg`*

Imports a new dataset from a given vector layer and saves the project unless the global option `-nosave` is used.

Required parameters

- `name=dataname` : defines the name of the new dataset
- `file=vectorlayer` : file containing the vector layer

After the `--dataset` command, the selected datasets correspond to the newly created one.

4.7.2 –distance_matrix : distance matrix of dataset or habitat

`--distance_matrix type=space|graph distance=leastcost|circuit|flow|circuitflow [dist=`*`val`* `proba=`

Required parameters

- `type`: distance calculation can be based on the euclidean or raster `space` or on the `graph`
- `distance`: from a raster, the distance can be `leastcost` or `circuit`, from a graph, the distance can be `leastcost`, `circuit`, `flow` or `circuitflow`

Optional parameters

For `flow` and `circuitflow` distances the following parameters must be given:

- `dist`: the distance between two patches
- `proba`: the probability of movement between two patches for the distance `dist`

Description

This command calculates distance matrix for each selected dataset (or habitat) and saves each matrix in a text file beginning with `distance_`.
For raster based distance, a matrix will be calculated for each selected dataset (or habitat) and linkset, for graph based distance, a matrix will be calculated for each selected dataset (or habitat) and graph.

Example

In the following example, the least cost distances between points are calculated based on cost raster of each selected linkset and for each selected pointset.

```
--distance_matrix type=space distance=leastcost
```

4.7.3 –model : calculate a model

`--model` *`variable`* `distW=`*`min:inc:maxcost`* `[vars=`*`var1,...,varn`*`] [raster=`*`r1,...,rn`*`]`

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

Optional parameters

- `vars=var1,...,varn` : adds other variables from the patch layer (Capacity for instance)
- `raster=r1,...,rn` : adds variables from external raster file

Description

Calculates a logistic model on each metric existing in all selected graphs for the binary dataset *variable*. The *variable* must exist in all selected datasets. For each graph, the command search for pointsets which have the same linkset as the graph. If none exists, it will throw an error. 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	Pointset	Metric	DistW	R2	p-value	Coef
2000m-3500cost	point-3500cost	BC_d3500_p0.05_beta1	1000.00	0.229898	0.00268	3.63230e-07
2000m-3500cost	point-3500cost	BC_d3500_p0.05_beta1	2000.00	0.134672	0.12585	4.89398e-08
2000m-3500cost	point-3500cost	F_d3500_p0.05_beta1	1000.00	0.522162	1.29e-5	0.00155784
2000m-3500cost	point-3500cost	F_d3500_p0.05_beta1	2000.00	0.266793	0.00074	0.000145906

References

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

4.8 Adding/Removal

4.8.1 –delta: remove one item

`--delta [filter=`*`condition`* `| [obj=patch|link [sel=`*`id1,id2,...,idn`* `| fsel=`*`file.txt`*`]]`

Optional parameters

- `filter=condition`: allow to filter graph element to test cf. Parameter filter=condition
- `obj=patch|link`: removal of patches (`obj=patch`) or links (`obj=link`)
- `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 a global metric in delta mode on nodes and/or links according to the `obj` parameter for each selected metric.

If `sel` or `fsel` is set, the calculation is performed only on the listed elements.
The result is saved in a patch and/or link attribute. The project is saved unless the global option `-nosave` is used.

Examples

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

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

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

```
--gmetric EC d=1000 p=0.05 --delta obj=patch sel=2,3
```

4.8.2 –gremove: remove several items

`--gremove `*`global_metric_name`* `[`*`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 (EC, PC, ...)

Optional parameters

- `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
```

4.8.3 –gtest: removal and iterative item addition

`--gtest` *`nstep global_metric_name`* `[`*`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

- `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 after removing the 8 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
```

4.8.4 –addpatch : iterative patch addition

`--addpatch npatch=`*val* `[hab=name] gridres=`*min:inc:max* `[capa=`*capa_file*`]`
`| patchfile=`*file.gpkg* `[capa=`*capa_field*`]`

Global paramaters

- `npatch` : number of patches to be added
- `hab=name`: name of the habitat in which the patches will be added if the metric has been calculated on a graph containing several habitats.

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 metric. Results are stored in a project subdirectory created for each metric.

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 or polygons from a vector layer.

Vector data test

`--addpatch npatch=`*val* `[hab=name] patchfile=`*file.gpkg* `[capa=`*capa_field*`]`

For each element of the vector layer (point or polygon), the program tests the addition of a patch on the pixel(s) covering the object, if the pixels are not NoData or already in the habitat class.

The `capa` parameter defines an attribute of the vector layer 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=`*val* `[hab=name] gridres=`*min:inc:max* `[capa=`*capa_file*`]`

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.

Examples

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

Adds 5 patches maximizing metric IIC, testing the addition of a patch every 100 meters. Results are stored in *addpatch_n5_IIC_graph_res100* subdirectory :

- addpatch_IIC_graph.shp : contains the added patches and the metric value
- addpatch_IIC_graph.txt : contains for each added patch the metric value
- links_IIC_graph.shp : contains the linkset of the final graph
- topo-links_IIC_graph.shp : contains the topological linkset of the final graph
- detail/ : subdirectory containing the detail of each test for each step
- detal/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.

```
--usemetric IIC_graph --addpatch npatch=5 gridres=100,200,500
```

Another example with a vector points set :

```
--usemetric IIC_graph --addpatch npatch=5 patchfile=testpoint.gpkg
```

Results are stored in *addpatch_n5_IIC_graph_vecttestpoint.gpkg* subfolder.

Limitations

The `--addpatch` command only works with graphs from complete topology linkset and without intra-patch distances.

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)]

4.8.5 –remelem: iterative item removal

`--remelem` *nstep global_metric_name* `[`*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.
- `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)]

4.9 Land use changes

4.9.1 –landmod : land use changes

`--landmod zone=`*`filezones.gpkg`* `id=`*`fieldname`* `code=`*`fieldname`* `[sel=`*`id1,...,idn`*`]`

Required parameters

- `zone=filezones.gpkg` : polygon layer containing land use changes.
- `id=fieldname` : field name of the layer 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 layer storing the new land use category of the polygon.

Optional parameter

- `sel=id1,...,idn` : list of polygon identifiers to process. If this parameter is not set, all polygons of the shapefile are processed.

Description

This command must be placed before other calculation commands. It will duplicate the project for each polygon of the polygon layer 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.

Exemple

```
--landmod zone=zones.gpkg id=id code=code --linkset distance=euclid --graph
--gmetric IIC
```

For each polygon identifier contained in the `zones.gpkg` layer, 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 craph. Finally, each subdirectory of the current project will contain a file `IIC.txt` containing the metric value taking into account each land use changes.

If the aim is to calculate the impact of changes using a global metric, it's easier to use the following command `--landmodgm`.

Reference

[Sahraoui et al.(2017)]

4.9.2 –landmodgm : land use changes on global metric

`--landmodgm zone=`*`filezones.gpkg`* `id=`*`fieldname`* `code=`*`fieldname`* `[sel=`*`id1,...,idn`*`]`

Required parameters

- `zone=filezones.gpkg` : polygon layer containing land-use changes
- `id=fieldname`: name of a layer attribute used to identify polygons. If the values are not unique, polygons with the same identifier will be applied in a single change.
- `code=fieldname` : name of a layer attribute storing the polygon's new land use category.

optional parameter

- |verb|sel=id1,...,idn| : list of polygon identifiers to be processed. If this parameter is not set, all polygons in the layer will be processed.

Description

This command will be executed on each global metric calculated or selected previously. It will duplicate the project for each polygon in the zone layer and modify the land use covered by the polygon(s). The link set and the graph for the selected metric will be created in each new project, along with the global metric.

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

A `.csv` file will be created storing the results for each global metric.

Example

```
--linkset distance=euclid --graph --gmetric IIC
--landmodgm zone=zones.gpkg id=id code=code
```

For each polygon identifier contained in the `zones.gpkg` layer, a new project named by the value of `id` will be created in a subdirectory of the current project. On each of these projects, the IIC metric link set and graph will be created, and the IIC metric calculated. Finally, a `landmod-iic.csv` file will be created, containing the value of the metric for each land use change.

4.10 Options

Global options start with only one dash and applied on the whole command line execution.

4.10.1 -nosave

This option prevents saving a project. It is useful when you don't want commands to modify the project.

4.10.2 -distconv

This option is used to set optional parameters for the distance conversion (ie. when distance is enclosed with braces). It contains only one parameter for the moment: `excost=val`, for excluding paths containing costs greater or equal than `val` from the regression. It is useful to exclude paths crossing barrier elements from the distance conversion.

4.10.3 -proc

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

4.10.4 -mpi

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

Chapter 5

Command examples

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

5.1 Project creation up to the graph

Create a new project named `sample` in the current directory from the `os.tif` landscape map.

```
java -jar graphab-3.0.jar --create sample os.tif
```

Adds a habitat to the `sample` project named `fields`, the habitat patches correspond to code 8 on the landscape map.

```
java -jar graphab-3.0.jar --project sample/sample.xml --habitat name=fields codes=8
```

Creates a link set `fields_euclid` in euclidean distance and planar topology based on the habitat `fields`.

```
java -jar graphab-3.0.jar --project sample/sample.xml --linkset distance=euclid
name=fields_euclid
```

Creates a graph `g_fields_euclid_500` pruned to 500m.

```
java -jar graphab-3.0.jar --project sample/sample.xml --graph name=g_fields_euclid_500
threshold=500
```

All the above commands can be executed at once:

```
java -jar graphab-3.0.jar --create sample os.tif --habitat name=fields codes=8
--linkset distance=euclid name=fields_euclid
--graph name=g_fields_euclid_500 threshold=500
```

5.2 Habitats

Creates a new habitat `fields1a` whose patches must be at least 1 are (0.01 hectare).

```
java -jar graphab-3.0.jar --project sample/sample.xml --habitat name=fields1a codes=8
minarea=0.01
```

The same process can be carried out starting from the `fields` habitat, eliminating spots with a capacity $< 100m^2$.

```
java -jar graphab-3.0.jar --project sample/sample.xml --vhabitat name=fields1a
from=fields mincapa=100
```

Creates another habitat `forest` from land use codes 3 and 4. Patches exceeding 500 meters in length or width are split up.

```
java -jar graphab-3.0.jar --project sample/sample.xml --habitat name=forest codes=3,4
maxsize=500
```

5.3 Link sets and graphs

5.3.1 Link sets

Creates a link set `fields_cost` from the habitat `fields` in cost distance and planar topology.

```
java -jar graphab-3.0.jar --project sample/sample.xml --usehabitat fields
--linkset distance=cost name=fields_cost 8=1 2=3 3,4=10 0,6=100 1,5=1000
```

Creates a link set `fields_comp_cost` from the habitat `fields` in cost distance and complete topology. In complete topology, it's important to specify a maximum distance, in this case 1000 cost units.

```
java -jar graphab-3.0.jar --project sample/sample.xml --usehabitat fields
--linkset distance=cost topo=complete maxcost=1000 name=fields_comp_cost
8=1 2=3 3,4=10 0,6=100 1,5=1000
```

5.3.2 Graphs

Creates a graph `g_fields_comp_cost` based on the link set `fields_comp_cost`.

```
java -jar graphab-3.0.jar --project sample/sample.xml --uselinkset fields_comp_cost
--graph name=g_fields_comp_cost
```

Creates a graph `g_fields_comp_cost_500` based on the link set `fields_comp_cost` and pruned to a cost distance equivalent to 500 meters. Distance conversion is performed automatically: 500 meters is converted in this case to 261.5 cost units.

```
java -jar graphab-3.0.jar --project sample/sample.xml --uselinkset fields_comp_cost
--graph name=g_fields_comp_cost_500 threshold={500}
```

5.4 Metrics

5.4.1 Global metrics

Global metric with several distances

Calculates the EC metric on the unpruned graph `g_fields_comp_cost` by varying the parameter `d` from 100 to 900 in steps of 200:

```
java -jar graphab-3.0.jar --project sample/sample.xml
--usegraph g_fields_comp_cost --gmetric EC d=100:200:900 p=0.05
```

The result is stored in the EC.txt file in the project directory:

Graph	d	p	EC
g_fields_comp_cost	100.0	0.05	301923.01540184487
g_fields_comp_cost	300.0	0.05	359617.35658238636
g_fields_comp_cost	500.0	0.05	409521.1200767072
g_fields_comp_cost	700.0	0.05	451094.0363294105
g_fields_comp_cost	900.0	0.05	486124.7711331516

Calculates the EC metric on the unpruned graph `g_fields_comp_cost` by varying the parameter `d` in metric distance (from 500m to 2000m) with automatic conversion to cost distance:

```
java -jar graphab-3.0.jar --project sample/sample.xml
--usegraph g_fields_comp_cost --gmetric EC d={500:500:2000} p=0.05
```

Le résultat est stocké dans le fichier EC.txt dans le répertoire du projet :

Graph	d	p	EC
g_fields_comp_cost	261.5279516552625	0.05	349052.7726285711
g_fields_comp_cost	486.5823280879073	0.05	406448.22457832936
g_fields_comp_cost	699.6560264253405	0.05	451028.7186860784
g_fields_comp_cost	905.3042342469763	0.05	486981.8241196714

5.5 Multi-habitat

Creates a link set `fields_forest_euclid` between habitat `fields` and habitat `forest` in euclidean distance and complete topology thresholded at 1000m. With the `inter` keyword, only links between two patches of different habitats are retained.

```
java -jar graphab-3.0.jar --project sample/sample.xml --mergehabitat fields,forest
--linkset distance=euclid topo=complete maxcost=1000 inter name=fields_forest_euclid
```

5.6 Show project details

Show project elements :

```
java -jar graphab-3.0.jar --project sample/sample.xml --show
```

Result:

```
===== Habitats =====
0 - fields
2 - fields1a
1 - forest

===== Linksets =====
fields_comp_cost
fields_cost
fields_euclid
fields_forest_euclid

===== Graphs =====
g_fields_comp_cost
g_fields_comp_cost_500
g_fields_euclid_500

===== Metrics =====
ec_d100_p0.05_g_fields_comp_cost
ec_d261.52795165526254_p0.05_g_fields_comp_cost
ec_d300_p0.05_g_fields_comp_cost
....
```

Show habitats details:

```
java -jar graphab-3.0.jar --project sample/sample.xml --show_habitats
```

Résultat :

```
===== Habitats =====
0 - fields
```

```
Habitat name : fields
# patches: 270

Patch connexity : 4-connexity
Habitat patch codes : [8]
Minimum patch area : 0.0 m2
Maximum patch size : 0.0 m

===== Habitats =====
2 - fields1a
Habitat name : fields1a
# patches: 68

Habitat from vector layer

===== Habitats =====
1 - forest
Habitat name : forest
# patches: 1,334

Patch connexity : 4-connexity
Habitat patch codes : [3, 4]
Minimum patch area : 0.0 m2
Maximum patch size : 500.0 m
```

Chapter 6

Processing capabilities and limitations

Graph-based methods provide an efficient modeling framework, but they can raise a question of computing capacity. Two specific points have received particular attention in Graphab: (1) calculation of link sets, (2) calculation of connectivity metrics. All these computations have been optimized by parallelization. This development mode improves computational efficiency by using a multi-processor architecture, a quad-core processor being theoretically four times faster than a single-core processor.

In [Foltête et al.(2012a)], several tests were conducted to measure the computational capacity of Graphab in different configurations. Three configurations were compared for these tests: (1) one core (3 Go RAM) corresponding to a current desktop computer, (2) four cores (6 Go RAM) corresponding to a workstation, and (3) 20 cores (15 Go RAM) corresponding to a server. The landscape map used was a grid of 14000*18000 pixels (252 millions of pixels) representing the landscape elements of the region of Franche-Comté (France) at a spatial resolution of 10 m. The landscape map contained 22,634 habitat patches.

Topology	Distance	Current desktop	Workstation	Server
Complete	Euclidean	1927s (32 min)	516s (8 min)	133s (2 min)
	Least-cost	19252s (5h 21 min)	4301s (1h 11 min)	1037s (17 min)
Planar	Euclidean	43s	12s	2.6s
	Least-cost	1080s (18 min)	295s (5 min)	82s (1 min)

Table 6.1: Computation times (seconds) required for calculating several link sets

The memory used by the software plays an important role. If there is not enough RAM, computation will be slower or may fail (OutOfMemoryError or GC Overhead message). The File/ Preferences / Memory menu can be used to adjust the memory allocated to Graphab. If you have a 32-bit version of Java, Graphab will be limited to about 2 Go (2000 Mo) of memory. If your computer has more than 2 Go of RAM memory, it is highly recommended you install the 64-bit version of Java to use the available memory beyond 2 Go.

6.1 Parallelism to speed up execution

6.1.1 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-3.0.jar -proc 8 --project path2myproject/myproject.xml ...
```

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

6.1.2 Computer cluster : mpi

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

```
mpirun java -jar graphab-3.0.jar -mpi --project path2myproject/myproject.xml ...
```

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

6.2 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-3.0.jar ... # 4Gb allocated
java -Xmx1500m -jar graphab-3.0.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.

Chapter 7

Metrics

Family	Connectivity metrics	Code	Computing level			Delta	Multi
			Global	Comp.	Local	metrics	habitat
Weighted	Flux	F	×	×	×		×
metrics	Equivalent Connectivity	EC	×	×		×	×
	Probability of Connectivity	PC	×	×		×	
	Interaction Flux (replace FPC)	IF			×		×
	Fractions of delta Probability of Connectivity	dPC				×	
	Betweenness Centrality index	BC			×		×
	Integral Index of Connectivity	IIC	×	×		×	
	Current Flow	CF			×		
Area	Mean Size of the Components	MSC	×				
metrics	Size of the Largest Component	SLC	×				
	Class Coincidence Probability	CCP	×				
	Expected Cluster Size	ECS	×				
Topological	Node Degree	Dg			×		
metrics	Clustering Coefficient	CC			×		
	Closeness Centrality	CCe			×		
	Eccentricity	Ec			×		
	Connectivity Correlation	CCor			×		
	Number of Components	NC	×				
	Graph Diameter	GD	×	×		×	
	Harary Index	H	×	×		×	
	Wilks' Lambda	W	×				

Table 7.1: Summary table of metrics in Graphab and computing level

Terms	Meaning
n	Number of patches
nc	Number of components
n_k	Number of patches in the component k
N_i	All patches close to the patch i
a_i	Capacity of the patch i (generally the surface area)
ac_k	Capacity of the component k (sum of the capacity of the patches composing k)
A	Area of the study zone
d_{ij}	Distance between the patches i and j (generally the least-cost distance between them)
$e^{-\alpha d_{ij}}$	Probability of movement between the patches i and j
α	Brake on movement distance
β	Exponent to weight more or less capacity

Table 7.2: Mathematical terms used

7.1 Weighted metrics

7.1.1 Flux

Flux (F)	Formula	Meaning
Global level	$S\#F = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} a_j^{\beta} e^{-\alpha d_{ij}}$	For the entire graph: sum of potential dispersions from all patches.
Local level	$F_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} a_j^{\beta} e^{-\alpha d_{ij}}$	For the focal patch i : sum of capacity of patches other than i and weighted according to their minimum distance to the focal patch through the graph. This sum is an indicator of the potential dispersion from the patch i or, conversely to the patch i.
Values	Values depend on the definition of a. If a represents an area, F expresses an area. Minimum value: 0 Maximum value: Total area of habitat if $\beta = 1$	
Comment	The path used in the graph is the one that maximizes $e^{-\alpha d}$, i.e. the one that minimizes the distance d (or the cost) between the patches i and j. This metric is called Area Weighted Flux (AWF) in some publications. However in Graphab, a is more general because it represents patch capacity, which may be their area or some other criterion chosen by the user. Similarly, the weighting is variable depending on the β parameter. In CS22, AWF is calculated only from patches directly connected to the focal patch, while Graphab takes into account indirectly connected patches.	
References	[Urban and Keitt(2001)] [Saura and Torné(2009)] [Foltête et al.(2012b)]	

7.1.2 Equivalent Connectivity

Equivalent Connectivity (EC)	Formula	Meaning
Global level Component level Delta	$EC = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j e^{-\alpha d_{ij}}}$	Square root of the sum of products of capacity of all pairs of patches weighted by their interaction probability.
Values	Unit corresponds to the capacity unit of the patches. Minimum value: 0 Maximum value: sum of capacities	
Comment	For each pair of patches, the path of the graph used is the one that maximizes $e^{-\alpha d}$, i.e. the one that minimizes the distance d (or the cost) between the patches i and j.	
Reference	[Saura et al.(2011)]	

7.1.3 Probability of Connectivity

Probability of Connectivity (PC)	Formula	Meaning
Global level Component level Delta	$PC = \frac{1}{A^2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j e^{-\alpha d_{ij}}$	Sum of products of capacity of all pairs of patches weighted by their interaction probability, divided by the square of the area of the study zone. This ratio is the equivalent to the probability that two points randomly placed in the study area are connected.
Values	Values correspond to a probability. Minimum value: 0 Maximum value: 1	
Comment	For each pair of patches, the path of the graph used is the one that maximizes $e^{-\alpha d}$, i.e. the one that minimizes the distance d (or the cost) between the patches i and j. The metrics PC and EC are linked by this equality : $PC = \left(\frac{EC}{A}\right)^2$ This metric is not available when a_i does not represent patch area.	
Reference	[Saura and Pascual-Hortal(2007)]	

7.1.4 Interaction Flux (replace FPC)

Interaction Flux (IF)	Formula	Meaning
Local level	$IF_i = \sum_{j=1}^{n} a_i^{\beta} a_j^{\beta} e^{-\alpha d_{ij}}$	Sum of products of the focal patch capacity with all the other patches, weighted by their interaction probability.
Values	Minimum value: 0 Maximum value: sum of capacities	
Comment	For each pair of patches, the path of the graph used is one that maximizes $e^{-\alpha d}$, i.e. one that minimizes the distance d (or the cost) between the patches i and j. This metric corresponds to the local contribution of a patch in the PC index, since $PC = \frac{1}{A^2} \sum_i IF_i$, with $\beta = 1$, and also EC : $EC^2 = \sum_i IF_i$. It is the equivalent of $dPC_{i,flux}/2 + dPC_{i,area}$ within one factor. However, the IF metric is obtained faster because it is not calculated on the basis of patch removal (delta mode). $$IF_i = EC^2 \left(\frac{1}{2} dPC_{i,flux} + dPC_{i,area} \right)$$ We can also relate IF to the local metric F : $$IF_i = a_i^2 + a_i F_i$$	
References	[Foltête et al.(2014)],[Sahraoui et al.(2017)]	

7.1.5 Fractions of delta Probability of Connectivity

Fractions of delta Probability of Connectivity (dPC_i, $dPC_{i,area}$, $dPC_{i,flux}$, $dPC_{connector}$)		
	Formula	Meaning
Delta	$dPC_i = \frac{(PC - PC_i')}{PC} = dPC_{i,area} + dPC_{i,flux} + dPC_{i,connector}$ $dPC_{i,area} = \frac{a_i^2}{EC^2}$ $dPC_{i,flux} = 2\frac{IF_i - a_i^2}{EC^2}$	Rate of variation between the value of PC index and the value of PC' corresponding to the removal of the patch i. The value of dPC_i is decomposed into three parts: - $dPC_{i,area}$ is the variation induced by the area lost after removal; - $dPC_{i,flux}$ is the variation induced by the loss of interaction between the patch i and other patches; - $dPC_{i,connector}$ is the variation induced by the modification of paths connecting other patches and initially routed through i.
Values	Minimum value: 0 Maximum value: 1	
Comment	If a does not represent patch area, dPC_{area} does not express a loss of area but a loss of capacity.	
Reference	[Saura and Rubio(2010)]	

7.1.6 Betweenness Centrality index

Betweenness Centrality index (BC)	Formula	Meaning
Local level	$BC_i = \sum_j \sum_k a_j^\beta a_k^\beta e^{-\alpha d_{jk}}$ $j,k \in \{1..n\}, k < j, i \in P_{jk}$	Sum of the shortest paths through the focal patch i, each path is weighted by the product of the capacities of the patches connected and of their interaction probability. Pjk represents all the patches crossed by the shortest path between the patches j and k.
Values	Values depend on the configuration. They correspond to a weight of potential transit. Minimum value: 0 Maximum value: square of the total area of habitat.	
Comment	With an adjustment of $\alpha = 0$ and $\beta = 0$ (uniform weighting of paths), the BC index is the same as that used in other types of graphs. An adjustment of $\alpha = 0$ and $\beta = 1$ gives paths a weight proportional to the product of the capacities of the patches that they connect, whatever their distance. In [Foltête et al.(2012a), Foltête et al.(2012b)], the BC_l index was proposed so as to give greater weight to paths exceeding a given criterion (e.g. dispersal distance). But tests showed that this index was strongly correlated with the weighted BC adjusted with $\alpha = 0$. In [Bodin and Saura(2010)], the BC_{pc} is the weighted BC with d equal to the dispersal distance, α as $e^{-\alpha d} = 0.05$ and $\beta = 1$.	
Reference	[Bodin and Saura(2010)] [Foltête et al.(2012a)]	

7.1.7 Integral Index of Connectivity

Integral Index of Connectivity (IIC)	Formula	Meaning
Global level Component level Delta	$IIC = \frac{1}{A^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{a_i a_j}{1 + nl_{ij}}$	For the entire graph: product of patch capacities divided by the number of links between them, the sum is divided by the square of the area of the study zone. IIC is built like the PC index but using the inverse of a topological distance rather than a negative exponential function of the distance based on the link impedance.
Values	Minimum value: 0 Maximum value: 1	
Reference	[Pascual-Hortal and Saura(2006)]	

7.1.8 Current Flow

Current Flow (CF)	Formula	Meaning
Local level	$CF_i = \sum_j^n c_i^j$	Sum of currents passing through the patch i. c_i^j represents the current through the patch i when currents are sent from all patches (except j) to the patch j. The patch j is connected to the ground.
Values	Minimum value : 0 Maximum value : $(n-1)(n-2)$ if $\beta = 0$ $(n-2)\sum_i^{n-1} a_i$ if $\beta = 1$	
Comment	The CF metric uses the electrical circuit theory. Each link of the graph corresponds to a resistor, the current sources and the ground are attached to the patches. If $\beta = 0$, each patch emits a current of 1, if $\beta = 1$, each patch emits a current equals to its capacity. This metric can be seen as an equivalent of BC metric (with $\alpha = 0$ and $\beta = 0$) that considers all possible paths, not just the shortest one.	
Reference	[Girardet et al.(2015)]	

7.2 Area metrics

7.2.1 Mean Size of the Components

Mean Size of the Components (MSC)	Formula	Meaning
Global level	$MSC = \frac{1}{nc}\sum_{k=1}^{nc} ac_k$	Mean of the component capacities.
Values	Minimum value: minimum capacity Maximum value: SLC	

7.2.2 Size of the Largest Component

Size of the Largest Component (SLC)	Formula	Meaning
Global level	$SLC = max\{ac_k\}$	Largest capacity of components.
Values	Minimum value: minimum capacity Maximum value: maximum capacity	

7.2.3 Class Coincidence Probability

Class Coincidence Probability (CCP)	Formula	Meaning
Global level	$CCP = \sum_{k=1}^{nc} \left(\frac{ac_k}{\sum_l ac_l} \right)^2$	Probability that two points randomly placed on the graph belong to the same component.
Values	Minimum value: $\to 0$ (as many components as patches and regular capacities) Maximum value: 1 (only one component)	
Reference	[Pascual-Hortal and Saura(2006)]	

7.2.4 Expected Cluster Size

Expected Cluster Size (ECS)	Formula	Meaning
Global level	$ECS = \frac{1}{\sum_k ac_k} \sum_{k=1}^{nc} ac_k^2$	Expected size of a component
Values	Minimum value: minimum capacity (as many components as patches and regular capacities) Maximum value: sum of capacities (only one component)	
Reference	[O'Brien et al.(2006)]	

7.3 Topological metrics

7.3.1 Wilks' Lambda

Wilks' Lambda (W)	Formula	Meaning
Global level	$W_{Lambda} = \frac{\lvert W \rvert}{\lvert T \rvert}$	Ratio between within-classes (within-components) (co)variance $\lvert W \rvert$ and total (co)variance $\lvert T \rvert$
Values	Minimum value: 0 the patches belonging to the same component are identical for all the variables (*i.e.* perfect partition) Maximum value: 1 patches belonging to the same component differ as much as with the other patches (*i.e.* meaning an irrelevant partition)	
Reference	[Everitt and Dunn(2001)], [Foltête and Vuidel(2017)]	

7.3.2 Harary Index

Harary Index (H)	Formula	Meaning
Global level Component level Delta	$H = \frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n}\frac{1}{nl_{ij}}$	Sum of the inverse of the number of links between all pairs of patches.
Values	Minimum value: 0 for a graph with no edges Maximum value: $\frac{n(n-1)}{2}$ for a complete graph	
Comment	For pairs of patches not connected by a path, we have : $nl_{ij} = +\infty$	
Reference	[Ricotta et al.(2000)]	

7.3.3 Graph Diameter

Graph Diameter (GD)	Formula	Meaning
Global level Component level Delta	$GD = \underset{ij}{max} d_{ij}$ $GD = \underset{i}{max} Ec_i$	Greatest distance between two patches of the graph.
Values	Minimum value: 0 Maximum value: $+\infty$	
Comment	When the nodes i and j are not related $d_{ij} = 0$ This metric is the global version of the metric Ec_i	
Reference	[Urban and Keitt(2001)]	

7.3.4 Number of Components

Number of Components (NC)	Formula	Meaning
Global level	$NC = nc$	Number of components of the graph.
Values	Minimum value: 1 Maximum value : n	
Reference	[Urban and Keitt(2001)]	

7.3.5 Node Degree

Node Degree (Dg)	Formula	Meaning
Local level	$Dg_i = \lvert N_i \rvert$	Number of edges connected to the node i *ie.* number of patches connected directly to the patch i
Values	Minimum value: 0 Maximum value: n	
Comment	There is an equivalence between the node degree and the number of nearest patches because graphs are not oriented and do not contain any loops.	
Reference	Freeman, 1979	

7.3.6 Clustering Coefficient

Clustering Coefficient (CC)	Formula	Meaning
Local level	$CC_i = \frac{1}{\lvert N_i \rvert (\lvert N_i \rvert - 1)} \sum_{j \in N_i} \lvert N_i \cap N_j \rvert$	Ratio of the number of nodes close to i which are neighbors to each other over the possible total.
Values	Minimum value: 0 Maximum value: 1	
Comment	Si $\lvert N_i \rvert \leq 1 \rightarrow CC_i = 0$	
Reference	[Ricotta et al.(2000)]	

7.3.7 Closeness Centrality

Closeness Centrality (CCe)	Formula	Meaning
Local level	$CCe_i = \frac{1}{n_k - 1} \sum_{\substack{j=1 \\ j \neq i}}^{n_k} d_{ij}$	Mean distance from the patch i to all other patches of its component k.
Values	Minimum value: 0 Maximum value: $+\infty$	
Comment	Si $n_k = 1 \rightarrow CCe_i = 0$	
Reference	Freeman, 1979	

7.3.8 Eccentricity

Eccentricity (Ec)	Formula	Meaning
Local level	$Ec_i = \max_j d_{ij}$	Maximum distance from the patch i to another patch of its component k.
Values	Minimum value: 0 Maximum value: $+\infty$	
Reference	[Urban and Keitt(2001)]	

7.3.9 Connectivity Correlation

Connectivity correlation ($CCor$)	Formula	Meaning
Local level	$CCor_i = \frac{\lvert N_i \rvert^2}{\sum_{j \in N_i} \lvert N_j \rvert}$	Ratio between the degree of the node i and the degree of its neighboring patches j
Values	Minimum value: 0 Maximum value: $\lvert N_i \rvert$	
Comment	If $\lvert N_i \rvert = 0 \rightarrow CCor_i = 0$	
Reference	[Minor and Urban(2008)]	

Bibliography

[Bodin and Saura(2010)] Orjan Bodin and Santiago Saura, 2010. Ranking individual habitat patches as connectivity providers: Integrating network analysis and patch removal experiments. *Ecological Modelling*, 221(19) 2393 – 2405.

[Brandes et al.(2008)] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner, 2008. On modularity clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 20(2) 172–188.

[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.

[Everitt and Dunn(2001)] Brian S Everitt and Graham Dunn, 2001. *Applied multivariate data analysis*, volume 2. Wiley Online Library.

[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.(2021)] Jean-Christophe Foltête, Gilles Vuidel, Paul Savary, Céline Clauzel, Yohan Sahraoui, Xavier Girardet, and Marc Bourgeois, 2021. Graphab: an application for modeling and managing ecological habitat networks. *Software Impacts*, 8 100065.

[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.(2015)] Xavier Girardet, Géraldine Conruyt-Rogeon, and Jean-Christophe Foltête, 2015. Does regional landscape connectivity influence the location of roe deer roadkill hotspots? *European Journal of Wildlife Research*, 61(5) 731–742.

[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.

[McRae et al.(2008)] Brad H. McRae, Brett G. Dickson, Timothy H. Keitt, and Viral B. Shah, 2008. Using circuit theory to model connectivity in ecology, evolution, and conservation. *Ecology*, 89(10) 2712 – 2724.

[Minor and Urban(2008)] Emily Minor and Dean Urban, 2008. A graph-theory framework for evaluating landscape connectivity and conservation planning. *Conservation biology*, 22 297–307.

[Newman(2006)] M. E. J. Newman, 2006. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23) 8577–8582.

[O'Brien et al.(2006)] Dan O'Brien, Micheline Manseau, Andrew Fall, and Marie-Josée Fortin, 2006. Testing the importance of spatial configuration of winter habitat for woodland caribou: An application of graph theory. *Biological Conservation*, 130(1) 70 – 83.

[Pascual-Hortal and Saura(2006)] Lucía Pascual-Hortal and Santiago Saura, 2006. Comparison and development of new graph-based landscape connectivity indices: towards the priorization of habitat patches and corridors for conservation. *Landscape Ecology*, 21(7) 959–967.

[Ricotta et al.(2000)] C. Ricotta, A. Stanisci, G.C. Avena, and C. Blasi, 2000. Quantifying the network connectivity of landscape mosaics: a graph-theoretical approach. *Community Ecology*, 1(1) 89–94.

[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.

[Saura et al.(2011)] Santiago Saura, Christine Estreguil, Coralie Mouton, and Mónica Rodríguez-Freire, 2011. Network analysis to assess landscape connectivity trends: Application to european forests (1990–2000). *Ecological Indicators*, 11(2) 407 – 416.

[Saura and Pascual-Hortal(2007)] Santiago Saura and Lucía Pascual-Hortal, 2007. A new habitat availability index to integrate connectivity in landscape conservation planning: Comparison with existing indices and application to a case study. *Landscape and Urban Planning*, 83(2–3) 91 – 103.

[Saura and Rubio(2010)] Santiago Saura and Lidón Rubio, 2010. A common currency for the different ways in which patches and links can contribute to habitat availability and connectivity in the landscape. *Ecography*, 33(3) 523–537.

[Saura and Torné(2009)] Santiago Saura and Josep Torné, 2009. Conefor sensinode 2.2: A software package for quantifying the importance of habitat patches for landscape connectivity. *Environmental Modelling and Software*, 24(1) 135 – 139.

[Urban and Keitt(2001)] Dean Urban and Timothy Keitt, 2001. Landscape connectivity: A graph-theoretic perspective. *Ecology*, 82(5) 1205–1218.