Reference Manual
Graphab 1.2
Command Line Interface

March 11, 2014

Contents

Chapter 1

Prerequisite

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

1.1 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-1.2.jar --help
```

Result

```
Usage :
java -jar graphab.jar --project prjfile.xml [--proc n]
...
...
```

You're ready to use Graphab in CLI mode !

1.2 Syntax

1.2.1 Definition

Commands always start with a double dash.
A global option starts with only one dash.
A parameter does not have a dash.

1.2.2 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.

1.2.3 Optional parameter

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

1.2.4 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.

1.2.5 Command execution

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

```
java -jar Graphab.jar --project prj.xml --graph --gmetric NC
```

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

Chapter 2

Command reference

2.1 General command

2.1.1 –help : display help

Command :

```
java -jar graphab-1.2.jar --help
```

Result :

```
Usage :
java -jar graphab.jar --metrics
java -jar graphab.jar --project prjfile.xml [-proc n] [-nosave] command1 [command2 ...]
Commands list :
--show
--linkset [complete[=dmax]] [code1,..,coden=cost1 ...] codei,..,codej=min:inc:max
--uselinkset linkset1,...,linksetn
--graph [nointra] [threshold=min:inc:max]
--usegraph graph1,...,graphn
--pointset pointset.shp
--usepointset pointset1,...,pointsetn
--gmetric global_metric_name [maxcost=valcost] [param1=min:inc:max [param2=min:inc:max ...]]
--cmetric comp_metric_name [maxcost=valcost] [param1=min:inc:max [param2=min:inc:max ...]]
--lmetric local_metric_name [maxcost=valcost] [param1=min:inc:max [param2=min:inc:max ...]]
--model variable distW=min:inc:max
--delta global_metric_name [maxcost=valcost] [param1=val ...] obj=patch|link [sel=id1,id2,...,idn]
--gtest nstep global_metric_name [maxcost=valcost] [param1=val ...] obj=patch|link sel=id1,id2,...
--ltest nstep local_metric_name [maxcost=valcost] [param1=val ...] obj=patch|link sel=id1,id2,...
--gremove global_metric_name [maxcost=valcost] [param1=val ...] [patch=id1,id2,...,idn|fpatch=...
--addpatch npatch global_metric_name [param1=val ...] [gridres=min:inc:max [capa=capa_file]

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

2.1.2 –metrics : list available metrics

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

```
java -jar graphab-1.2.jar --metrics
```

Result :

```
===== Global metrics =====
S#F -  Flux (F)
       params : d, p, beta
```

```
PC - Probability of Connectivity (PC)
       params : d, p, beta
IIC - Integral index of connectivity (IIC)
CCP - Class coincidence probability (CCP)
MSC - Mean size of the components (MSC)
SLC - Size of the largest component (SLC)
ECS - Expected Cluster Size (ECS)
GD - Graph diameter (GD)
H - Harary index (H)
NC - Number of components (NC)
dPC - Delta PC decomposed (dPC)
       params : d, p, beta

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

2.1.3 –project : load project

This command sets the path to the project xml file.

```
java -jar graphab-1.2.jar --project path2myproject/myproject.xml
```

This command loads the project myproject contained in the path2myproject folder. You cannot create a project in CLI mode, you must to create it beforehand in the GUI. The *–project* command can be used only once and must be the first command. All of the following commands below require a loaded project.

2.1.4 –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-1.2.jar --project path2myproject/myproject.xml --show
```

Result :

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

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

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

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

2.2 Graph management

2.2.1 –linkset : create cost linkset

```
--linkset [complete[=dmax]] [code1,..,coden=cost1 ...] codei,..,codej=min:inc:max
```

Creates a new linkset in the loaded project and saves the project unless the *-nosave* option is used. For the moment, this command does not permit using either euclidean distance or an external cost surface. The linkset's name is derived from the cost definition.
This command will create a planar linkset *cost_1_2_3_4_5_6_7-1.0* with all costs equal to 1 :

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

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

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

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

```
--linkset 4,5,6,7=10 1,2,3=100:50:200
or
--linkset 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.
The *–linkset* command does not accept several ranges.
After *–linkset* command execution, ensuing linkset selection is set to created linksets.

2.2.2 –uselinkset : select linkset

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

Selects linksets to be used in following commands.
The linkset name is case sensitive and must not contain blank spaces.
By default, all linksets are selected.

2.2.3 –graph : create graph

```
--graph [nointra] [threshold=min:inc:max]
```

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

```
--graph
```

The graph name will be the concatenation of *comp_* and linkset name.
If a unique value threshold is specified, the command will create one graph with the given threshold for each selected linkset :

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

The graph name will be the concatenation of *thresh_100.0_* and the linkset name.
If the threshold parameter is defined with a value list or a range, it will create a thresholded graph for each linkset and each threshold :

```
--graph threshold=1000:100:1500
or
--graph threshold=1000,1100,1200,1300,1400,1500
```

Result : this command created 6 graphs for each selected linkset.
The *nointra* option disables intra-patch distances for this graph.
After *–graph* command execution, ensuing graph selection is set to the created graphs.

2.2.4 –usegraph : select graph

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

Selects graphs to be used in following commands.
The graph name is case sensitive and must not contain blank spaces.
By default, all graphs are selected.

2.3 Calculate metric

2.3.1 –gmetric : calculate global metric

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

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.
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
```

If the *maxcost* parameter is defined, paths greater than *maxcost* are not calculated. It can reduce the time execution for path-based metric but results may be inaccurate.

2.3.2 –cmetric : calculate component metric

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

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

If the *maxcost* parameter is defined, paths greater than *maxcost* are not calculated. It can reduce the time execution for the path-based metric but the result may be inaccurate.

2.3.3 –lmetric : calculate local metric

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

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

If the *maxcost* parameter is defined, paths greater than *maxcost* are not calculated. It can reduce the time execution for the path-based metric but the result may be inaccurate.

2.4 SDM

2.4.1 –pointset : import pointset

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

Creates a new pointset from the shapefile parameter for each selected linkset and saves the project unless *-nosave* option is used. The pointset name is a concatenation of the shapefile name and the linkset name. After *–pointset* command execution, pointset selection is set to the created pointsets only.

2.4.2 –usepointset : select pointset

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

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

2.4.3 –model : calculate SDM

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

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

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

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

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

2.5 Adding/Removal

2.5.1 –delta

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

Calculates global metric in delta mode on patches or links depending on *obj* parameter for each selected graph. If the *sel* parameter is defined, it calculates only for elements listed in *sel*. Results are stored in a text file for each graph in the project folder. The file name is the concatenation of 'delta-' + metric short name + graph name.
To execute the NC metric in delta mode for each patch :

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

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

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

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

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

Init corresponds to the initial PC value, without removing element. Following values correspond to the relative loss of the metric value when removing the element from the graph. Removing the patch with id equal to 2, the PC decreases by 0.269%
Ranges are not allowed in delta metric parameters.
If the *maxcost* parameter is defined, paths greater than *maxcost* are not calculated. It can reduce the time execution for the path-based metric but the result may be inaccurate.

2.5.2 –gremove

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

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

Results are only displayed. For each graph, graphab show 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 id does not exist, it will be ignored.
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
```

Ranges are not allowed in metric parameters.
If the *maxcost* parameter is defined, paths greater than *maxcost* are not calculated. It can reduce the time execution for the path-based metric but the result may be inaccurate.

2.5.3 –gtest

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

Not yet documented
Ranges are not allowed in metric parameters.
If the *maxcost* parameter is defined, paths greater than *maxcost* are not calculated. It can reduce the time execution for the path-based metric but the result may be inaccurate.

2.5.4 –ltest

```
--ltest nstep local_metric_name [maxcost=valcost] [param1=val ...] obj=patch|link
  sel=id1,id2,...,idn
```

Not yet documented
Ranges are not allowed in metric parameters.
If the *maxcost* parameter is defined, paths greater than *maxcost* are not calculated. It can reduce the time execution for the path-based metric but the result may be inaccurate.

2.5.5 –addpatch : adding patches

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

This command tests successively the adding of a new patch from a predefined set of points and retains the patch which maximizes the given global metric. This process is repeated as many times as the *npatch* parameter and for each selected graph. Results are stored in a project subdirectory created for each graph.
For each testing of a new patch, the graph is recomputed including the new patch and the links connecting this patch to the others, then the given metric is calculated on this new graph. When all the possible patches was tested, the one maximizing the metric is added to the project. This process is iterated until having *npatch* new patches in the project.
The created patches have a size of one pixel, if the corresponding land cover pixel is already habitat or is outside of the study area, the patch is skipped. Patches location to be tested can be given through a regular grid or a set of points from a shapefile.

Point set test

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

For each point of the shapefile, the program tests the addition of a patch on the pixel covering the point, if the pixel is not NoData or already in the habitat class.
The *capa* parameter defines an attribute of the shapefile containing a capacity value for each tested patch. If the *capa* parameter is not specified, the capacity of new patches will be 1.

Regular grid test

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

Tests patches addition on a regular grid. The size of the grid cell is set by *gridres*.
The *capa* parameter is used to define a raster file (TIFF or AsciiGrid format) giving a value of potential capacity at any point of the study area. If the capacity is zero, no patch will be tested at this position. The raster file can have a different resolution than the grid or the landscape map. If the *capa* parameter is not specified, the capacity of new patches will be 1.
The *multi* parameter is used to test the simultaneous addition of *nbpatch* patches, in a neighborhood of radius *size*gridres*.

Examples

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

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

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

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

Another example with a shapefile points set :

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

Results are stored in *addpatch_n5_graph_IIC_shptestpoint.shp* subfolder.

Limitations

The *–addpatch* command only works with graphs from complete topology linkset.
This command changes the number of patches in the project; executing commands after it can lead to inconsistencies. Therefore, it is not recommended to add commands after the command *–addpatch*.
Ranges are not allowed in metric parameters.

2.6 Options

2.6.1 -nosave

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

2.6.2 -proc

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

Chapter 3

Command examples

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

3.1 Display project

First, display project elements :

```
java -jar graphab-1.2.jar --project sample_project/Project.xml --show
```

Result :

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

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

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

3.2 Batch metric parameter

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

```
java -jar graphab-1.2.jar --project sample_project/Project.xml
  --usegraph 2000m-3500cost --gmetric PC d=1000:1000:5000 p=0.05 beta=1
```

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

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

3.3 Batch thresholded graph and global metric

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

```
java -jar graphab-1.2.jar --project sample_project/Project.xml --uselinkset Complete_cost
  --graph 2000:100:2500 --gmetric IIC
```

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

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

3.4 Complete SDM sequence

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

```
java -jar graphab-1.2.jar --project sample_project/Project.xml -nosave
  --uselinkset Planar_Euclid
  --graph
  --lmetric Dg --lmetric F d=1000 p=0.05 beta=1
  --pointset sample_project/Exo-Presence_absence.shp
  --model PRESENCE distW=1000
```

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

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

Chapter 4

Performance tuning

4.1 Parallelism to speed up execution

4.1.1 One computer : threads

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

```
java -jar graphab-1.2.jar --project path2myproject/myproject.xml -proc 8 ...
```

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

4.1.2 Computer cluster : mpi

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

```
mpirun java -jar graphab-1.2.jar --mpi --project path2myproject/myproject.xml ...
```

Only some commands can be used in mpi environments : –gmetric, –cmetric, –lmetric, –delta, –addpatch

4.2 Memory management

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

```
java -Xmx2g -jar graphab-1.2.jar ... # 2Gb allocated
java -Xmx1500m -jar graphab-1.2.jar ... # 1500 Mb -> 1.5Gb allocated
```

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

```
java -version
```

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