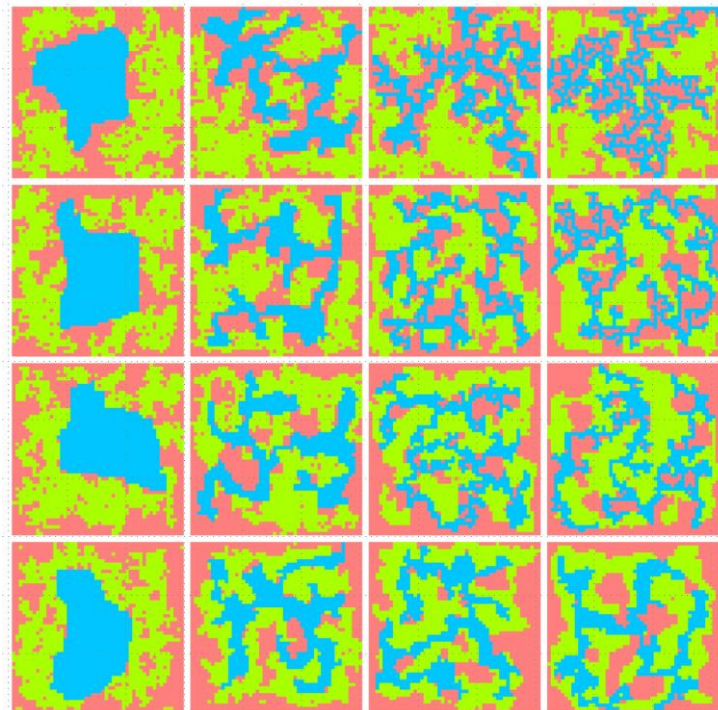


Landscape Generator (LG)

User's manual



Maarten J. van Strien, Cornelis T.J. Slager

Zurich, 30 March 2016

LG and its manual can be downloaded from: www.lg.ethz.ch

Disclaimer

The MIT License (MIT)

Copyright © 2016 Cornelis T.J. Slager & Maarten J. van Strien

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Citation

Please refer to Landscape Generator (LG) by citing the following two publications:

Slager CTJ, de Vries B (2013) Landscape generator: method to generate landscape configurations for spatial plan-making. *Computers Environment and Urban Systems*, 39, 1-11.

Van Strien MJ, Slager CTJ, de Vries B, Gret-Regamey A (in press) An improved neutral landscape model for re-creating real landscapes and generating landscape series for spatial ecological simulations. Ecology and Evolution

Contents

1. Introduction	4
2. Getting started	4
3. Configuration files.....	5
3.1 Criteria	7
3.2 Creating configuration files	8
4. Input rasters.....	9
4.1 Random percolation maps.....	9
4.2 Real landscape	10
5. Running LG	10
5.1 Single mode	10
5.2 Concurrent mode.....	10
6. References	11

1. Introduction

Landscape Generator (LG) is a program that generates raster maps of landscapes with a user-defined composition and configuration. Users can set target values of landscape metrics (criteria) that quantify landscape composition and configuration. These landscape metrics are comparable to the landscape metrics used in FRAGSTATS (McGarigal *et al.* 2012) and comprise both class-level (i.e. referring to the composition or configuration of an entire land-use class) and patch-level (i.e. referring to the composition or configuration of a single land-use patch) metrics.

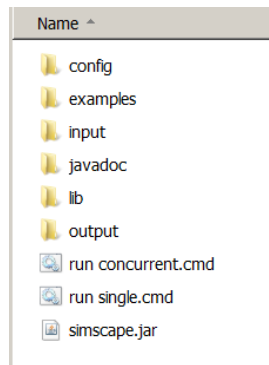
LG uses an optimisation algorithms to “find” landscapes of which the composition and configuration corresponds to the target values. It does so by iteratively swapping raster cells and determining whether the new landscape is an improvement with regard to the target values. For more details and information about the optimisation algorithms used in LG we refer to Slager (2011) and Slager and de Vries (2013). The fact that LG uses an optimisation algorithm means that (1) calculation times can vary depending on the size and complexity of the landscape and its configuration, (2) generation of a landscape can fail due to LG being stuck in a local optimum or (3) generation of a landscape can fail due to an incompatible set of target values (e.g. trying to generate a landscape in which a certain land-use class consists of 40 patches, while the total area of the respective land-use class is 20 raster cells). This means that, in some cases, LG has to be rerun to successfully generate a landscape or that the target values need to be redefined to speed up calculations or to reach a feasible landscape.

The input of landscape generator consists of (1) **configuration files** with target values for the different criteria or landscape metrics quantifying the landscape composition and configuration and (2) an **input raster** that can either be a random percolation map or an existing landscape to which you want to make changes. Below is explained how the target values and input raster have to be configured. The output of LG is either an **output raster** or, in case LG fails to generate a landscape, a **threshold-hit text file** showing the composition and configuration values just before LG stopped the process.

2. Getting started

LG is programmed in Java and thus runs on most operating systems (e.g. Windows, LINUX, Mac OS). Thus, before you can start using LG, Java (<https://java.com/en/download/>) and the Java Development Kit (<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>) should be installed on your computer. We tested LG on a computer with Windows 7 and Java version 8 update 20.

Once Java is installed, unzip the *LG.zip* file to any location on your computer. The newly created folder *LG* should contain the following files and folders:



The files *simscape.jar* and those in the folders *javadoc* and *lib* contain the Java-code of LG. These files do not need to be changed in order for LG to work, unless you are interested in changing the LG source code.

The *config*, *input* and *output* folders contain the in- and output of LG. The names of these folders can be changed, but we recommend to leave them as they are. In the folder *config* you should place the configuration file(s) with the target values for each landscape that you want to generate. Similarly, in the *input* folder you should place the input rasters that LG will use to generate new landscapes. Initially these two folders are empty, but below we describe how to create and format the files that should be placed in these folders. Output rasters (ASCII rasters and png-images) and threshold-hit text files will be placed in the *output* folder.

Lastly, the *examples* folder contains example configuration files and input rasters as well as example Python-scripts for generating these files. All the files in the *examples* folder were used to generate the results presented in Van Strien *et al.* (in press).

3. Configuration files

The configuration files are placed in the folder *config* and have the extension “.properties” (e.g. *landscape1.properties*). These configuration files are simple txt files that can be created or edited in a text-editor like Notepad and saved with the .properties extension. A user can create several configuration files and execute them all at once making use of all processor cores in a computer (see section “Running LG”).

Below is an example of a configuration file (i.e. the *in_example_1_ls_1.properties* configuration file in the *examples\config* folder):

```
input.file=lg_input_ls_1_example_1.asc
output.prefix=out_example_1_ls_1

improvement.threshold=2000000
parcel.basic.type=35.0,65.0
parcel.image.cell.size=1
parcel.image.lines=false
orderedcriteria.solveMode=sequentially

criteria.classes=ClusterCriteria 1,3;\
ClusterCriteria 2,1;\
GlobalPerimeterCriteria 1,246;\
AreaCriteria 1,0,33,1;\
AreaCriteria 1,1,33,1;\
AreaCriteria 1,2,33,1;
```

The lines highlighted in yellow should be present in every configuration file, while the lines highlighted in blue are variable and depend on the target values that are set for the landscape that is to be generated.

The lines highlighted yellow are described below.

input.file= Here the name of the input raster should be written. The rasters should have the ASCII format, so the extension of the raster file should be .asc (e.g. lg_input_ls_1_example_1.asc). The input raster file should be present in the *input* folder.

output.prefix= Specify a unique string from which the output raster or threshold-hit text file can be recognised (e.g. out_example_1_ls_1).

improvement.threshold= This value sets a maximum to the number of iterations (i.e. number of times raster cells are switched) in the optimisation without any improvement towards reaching the target values. If no improvements have been made after many iterations, it may indicate that the optimisation is stuck in a local optimum or that incompatible target values were specified (see section “1. Introduction”). If this threshold is reached, a threshold-hit text file is written to the output folder.

parcel.basic.type= These values indicate the percentage area of certain land-use classes in the generated landscape. They should be separated by a comma, but contain no spaces. There should be as many values as there are land-use classes (e.g. with two land-use classes the values could be 35.0,65.0). The order of these values is the same as the order of the used land-use class numbers sorted from low to high.

parcel.image.cell.size= This value is the cell size of the input raster. Some target values (e.g. edge length and area) are relative to this value (e.g. a maximum edge length of 2000 m with a cell size of 100 m will get a target value of 20).

parcel.image.lines= Specify `true` if grey lines should be drawn in between raster cells on the output png image. Specify `false` if no lines should be drawn. This setting has no effect on the output ascii raster.

orderedcriteria.solve mode= Specify `sequentially` if the optimisation process should optimise one target value after the other (in the order specified in `criteria.classes`). Specify `simultaneously` if all target values should be optimised at once. We strongly recommend using `sequentially` (sensu Slager & de Vries 2013).

criteria.classes= Here a list (highlighted in blue above) of optimisation criteria should be specified. Each optimisation criterion refers to a certain landscape metric for which a target value should be specified (see below). Each criterion should be separated by “;”. If `orderedcriteria.solve mode=sequentially`, then the order in which the criteria are specified can influence the speed and success of the landscape generation. It could, thus, be worthwhile to experiment with different orders of the optimisation criteria. In section “3.1 Criteria” is an overview of the criteria that can be selected.

3.1 Criteria

In the configuration file, the lines highlighted in blue refer to the list of optimisation criteria, with which the configuration of the generated landscape can be regulated. With these criteria the target values of the different landscape metrics can be set. There are several criteria that the user can choose from.

Below is a list of the available criteria. In the brackets behind each criteria name, are the setting variables that have to be specified for a criteria to function. As all the criteria in LG refer to the class or patch level, first you should specify the land-use class (*landUse*) and potentially the patch number (*clusterIndex*) to which the criteria should be applied. The number code of a land-use class should be the same as in the input map. Patches are numbered continuously from 0 to number of patches-1 starting in the top left corner of the landscape. In the description of the criteria below, it is specified whether the target values should be “int” or “double”, meaning that their value MUST be integer (e.g. 1) or CAN be written as a decimal (e.g. both 1 and 1.2 would be correct), respectively. At the end of the description of a criteria we give an example of how the criteria should be formatted in the configuration file. Other examples can be found in the configuration files in the *examples\config* folder.

ClusterCriteria(int landUse, int count): Class-level criterion. Specifies the number of patches for a certain land-use class (*count*). LG considers the four direct neighbouring raster cells to determine patches, so a diagonal connection is regarded as a separate patch. Equivalent to the NI criterion in Slager and de Vries (2013). Example: `ClusterCriteria 2,1`

AreaCriteria(int landUse, int clusterIndex, double percentage, int deviation): Patch-level criterion. With this criterion the area of a certain patch can be set (*percentage*). This area is defined as the percentage of the total area occupied by the respective land-use class. In order to make this criterion not too restrictive, a maximum percentage points of deviation from the target value can be set (*deviation*). We recommend allowing a small deviation (e.g. 1% point) from the target value (Slager & de Vries 2013). Equivalent to the relative AREA criterion in Slager and de Vries (2013). Example: `AreaCriteria 1,0,33.3,1`

ExactAreaCriteria(int landUse, int clusterIndex, int exactArea): Patch-level criterion. With this criterion the area of a certain patch can be set (*exactArea*). This area is defined as the exact area that a patch will occupy in the generated landscape and basically translates to the number of raster cells that a certain patch should occupy. Therefore, this value is relative to the `parcel.image.cell.size` (see section “3. Configuration files”). Equivalent to the AREA criterion in Slager and de Vries (2013). Example: `ExactAreaCriteria 1,0,40`

GlobalPerimeterCriteria(int landUse, int maxPerimeter): Class-level criterion. Specifies the maximum perimeter of all patches for a certain land-use class (*maxPerimeter*). This value is relative to the `parcel.image.cell.size` (see section “3. Configuration files”). Equivalent to the TE criterion in Slager and de Vries (2013). Example: `GlobalPerimeterCriteria 1,246`

PerimeterCriteria(int landUse, int clusterIndex, int maxPerimeter): Patch-level criterion. Specifies the maximum perimeter of a certain patch (*maxPerimeter*). This value is relative to the `parcel.image.cell.size` (see section “3. Configuration files”). Equivalent to the PERIM criterion in Slager and de Vries (2013). Example: `PerimeterCriteria 1,0,20`

TouchCriteria(int landUse, int clusterIndex, int otherLandUse, int minPercentage, int maxPercentage): Patch-level criterion. With this criterion you can regulate to what percentage a certain patch is touching (or neighbouring) raster cells of another land-use class. The target values for this criterion are defined as the minimum (*minPercentage*) and maximum (*maxPercentage*) percentage of raster cell edges of the respective patch that are touching the other land-use class (*otherLandUse*). The minimum and maximum values can be the same. Equivalent to the ECON criterion in Slager and de Vries (2013). Example: `TouchCriteria 1,0,2,15,15`

AspectRatioCriteria(int landUse, int clusterIndex, int widthRatio, int heightRatio): Patch-level criterion. With this criterion the shape of the bounding box of a patch (i.e. the smallest rectangle that can completely surround a patch) can be regulated. The ratio of the dimensions of the bounding box can be controlled with the target values (*widthRatio*, *heightRatio*). Although named width and height, these values do actually not set the horizontal and vertical dimensions of the bounding box. Instead, they set the ratio of the longest and shortest edge of the bounding box. Thus, `AspectRatioCriteria 1,0,10,1` is the same as `AspectRatioCriteria 1,0,1,10`. In both cases the longest edge of the bounding box of patch 0 is 10 times longer than the shortest edge. Equivalent to the ARBB criterion in Slager and de Vries (2013). Example: `AspectRatioCriteria 1,0,10,1`

RCCriteria(int landUse, int clusterIndex, double percentage, double deviation): Patch-level criterion. This criterion regulates the percentage of the total area within a bounding box occupied by a certain patch (*percentage*). The maximum percentage points of deviation from the target value can also be set (*deviation*). This criterion is especially useful in combination with the `AspectRatioCriteria`, with which the bounding box aspect ratio can be set (see above). Setting *percentage* to 100 means that the whole bounding box of a certain patch will be filled with the respective land-use class. Equivalent to the PBB criterion in Slager and de Vries (2013). Example: `RCCriteria 1,0,100,0`

RCFCriteria(int landUse, int clusterIndex, int widthRatio, int heightRatio, double deviation, double minPerc, double maxPerc): Patch-level criterion. This criterion combines the `AspectRatioCriteria` and `RCCriteria`. For a certain patch, the aspect ratio of the bounding box can be set (*widthRatio* and *heightRatio*; see above) together with the percentage points of maximum permissible deviation from this aspect ratio (*deviation*). Also the minimum and maximum percentage as to which the bounding box has to be occupied with cells of the respective patch should be set (*minPerc* and *maxPerc*). Example: `RCFCriteria 1,0,3,1,20,100,100`

RectangleCriteria(int landUse, int clusterIndex, int percentage, int deviation): Patch-level criterion. With this criterion a certain patch can be made rectangular up to a certain percentage (*percentage*). The maximum percentage points of deviation from the target value can also be set (*deviation*). Setting *percentage* to 100 will result in a patch that is completely rectangular. This criterion is very similar to the `RCCriteria`, but works better in combination with `AreaCriteria`. Example: `RectangleCriteria 1,0,100,0`

3.2 Creating configuration files

The configuration files can be created manually in a text editor (e.g. Notepad, Notepad++). It is important is that they are saved with the .properties extension in the *config* folder. The

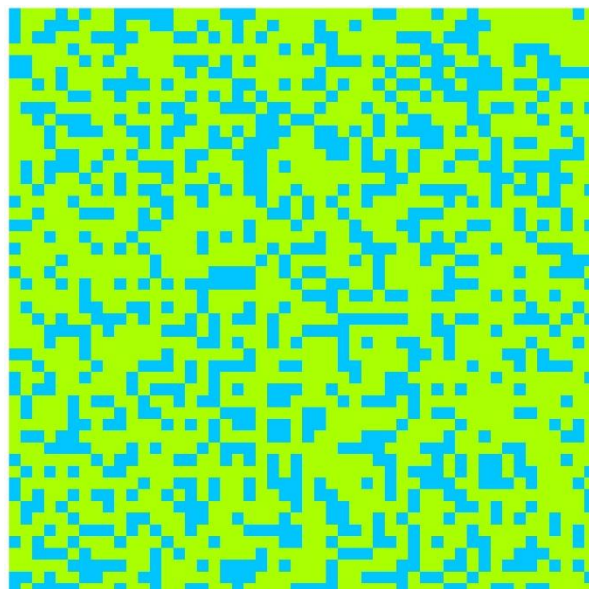
configuration files can also be generated in an automated way with any programming language (e.g. R, Python). This way, a multitude of configuration files can easily be generated with incrementally changing target values (e.g. Van Strien *et al.* in press). We have added two examples of Python-script files that have been used to generate the example configuration files in *examples\config*. These files are *Generate_LG_input_example_1.py* and *Generate_LG_input_example_2.py* and can be found in the *examples* folder. These scripts can be edited and executed in any Python IDE (e.g. PyScripter). ArcGIS 10.x is required to run these scripts. Generating such scripts to automatically write configuration files requires programming experience and is not further discussed in this user's manual.

4. Input rasters

The input rasters for LG should be in the ASCII format (.asc extension). This format can be opened, edited and stored in many GIS programs (e.g. ArcGIS, QGIS). Input rasters can either be a random percolation map or a map of an existing landscape (see below). The input rasters should be stored in the *input* folder.

4.1 Random percolation maps

If one is using LG to generate hypothetical landscapes that are not based on existing landscapes (e.g. Van Strien *et al.* in press), it is best to use random percolation maps as input raster. In random percolation maps each raster cell is randomly assigned a land-use class based on the proportion of occurrence of the different land-use classes. These proportions are thus the same as the proportions specified under `parcel.basic.type` in the configuration file (see section "3. Configuration files"). Below is an example of such a percolation map (i.e. *examples\input\lg_input_ls_1_example_1.asc*).



Random percolation maps can be created with several GIS and programming languages (e.g. R, Python). We have added several examples of the percolation map files that have been used to generate the example landscapes in *examples\input*. In the *examples* folder, the Python scripts *Generate_LG_input_example_1.py* and *Generate_LG_input_example_2.py* contain the

functions that we used to create these percolation maps. Exact instruction on how to create the percolation maps are beyond the scope of this user's manual.

4.2 Existing landscape

If LG is used to make changes to existing landscape configurations, then raster maps with these configurations can be used as input. Different land-use classes can be represented in the raster. These classes should be number coded (e.g. 1, 2, 3, 4). It is also possible to indicate raster cells that should not change to another land-use class by adding 100 to the land-use class code (e.g. a raster cell coded as 101 is classified as land-use class 1 and is supposed to remain class 1 during the generation of the landscape). Such input rasters can be created in most GIS programs (e.g. ArcGIS, QGIS).

5. Running LG

Once the configuration file(s) and input raster(s) have been created, LG can be run to generate the landscape(s). A single configuration file can be executed (single mode) or it is possible to let LG automatically execute all configuration files located in the *config* folder (concurrent mode). The latter mode will use all processor cores and automatically start generating the next landscape once a landscape is finished or reached the *improvement.threshold* (see section "3. Configuration files"). The files *run single.cmd* or *run concurrent.cmd* have to be edited to run in "single mode" or "concurrent mode", respectively. The successfully generated landscapes (both png and asc format) or threshold-hit text files will appear in the *output* folder.

5.1 Single mode

To run the "single mode" edit *run single.cmd* in a text editor (e.g. Notepad or Notepad++). The file will contain the following text:

```
java -jar simscape.jar single runX D:\LG\config\short_0.properties
D:\LG\output D:\LG\input
```

In this file change all three occurrences of *D:* to the path where the *LG* folder is located. Change *short_0.properties* to the name of the configuration file that you want to execute. Also change *config*, *output* and *input* if you have changed the name of these folders. Optionally, you can change *runX* to whatever you want to name the run of LG. Save the file. You should now be able to run LG by double clicking on *run single.cmd*. A box will appear showing you the progress of the landscape generation. Simply close this box, if you want to quit the program.

5.2 Concurrent mode

To run the "concurrent mode" edit *run concurrent.cmd* in a text editor (e.g. Notepad or Notepad++). The file will contain the following text:

```
java -jar simscape.jar concurrent runX D:\LG\config D:\LG\output D:\LG\input
```

In this file change all three occurrences of *D:* to the path where the *LG* folder is located. Also change *config*, *output* and *input* if you have changed the name of these folders. Optionally, you can change *runX* to whatever you want to name the run of LG. Save the file. You should

now be able to run LG by double clicking on *run concurrent.cmd*. A box will appear showing you the progress of the landscape generation. Simply close this box, if you want to quit the program.

6. References

- McGarigal K, Cushman SA, Ene E (2012) FRAGSTATS v4: spatial pattern analysis program for categorical and continuous maps, University of Massachusetts, Amherst, USA.
- Slager CTJ (2011) *Landscape generator - method to generate plausible landscape configurations for participatory spatial plan-making*. PhD thesis, Technical University Eindhoven, The Netherlands.
- Slager CTJ, de Vries B (2013) Landscape generator: method to generate landscape configurations for spatial plan-making. *Computers Environment and Urban Systems*, **39**, 1-11.
- Van Strien MJ, Slager CTJ, De Vries B, Gret-Regamey A (in press) An improved neutral landscape model for re-creating real landscapes and generating landscape series for spatial ecological simulations. *Ecology and Evolution*.