**LORACS – USER MANUAL – V 1.0 SEPTEMBER 2009**

**CONTENTS**

When LORACS is downloaded you should get the following:

1) loracs.jar: the binary file used to run LORACS.

2) doc: the directory containing the Java API for the methods used in LORACS. Useful only if you’re planning on modifying the program.

3) source: the directory containing the source code in Java. Useful only if you’re planning on modifying the program.

4) readme.doc: this manual.

5) pinto\_keitt2009.pdf: a Landscape Ecology paper showing the theoretical basis behind LORACS.

6) example: a directory containing example input/output files.

**REQUIREMENTS**

LORACS is platform independent Java software. In Windows/Linux it requires Java SDK 1.5 or higher, and in Mac OSX it requires Java SDK 1.6 or higher. Java JRE will not work. Java SDK can be downloaded for free from:

<http://www.java.com>

**HOW DO I OPEN THE PROGRAM?**

Windows GUI

1) Go to the directory where the jar file is saved.

2) Double click on the jar file.

Command line:

For clarity, all spaces are explicitly written in the command line instructions. Whenever you see [space] you should leave at least one space between expressions. In general you should use low /high caps as indicated here.

Windows command line:

1) Open the Command Prompt (Start --> Accessories --> Command Prompt)

2) Move to the directory where the jar file was saved. For example, let’s say the file was saved in the directory \Users\chris\Documents\my\_programs. Then type:

cd [space] \Users\chris\Documents\my\_programs

3) Make sure the jar file is in this directory by typing

dir

4) Start LORACS by typing

java [space] -jar [space] loracs.jar

If you wish to increase the memory allocated to the Java Virtual Machine (JVM), you’ll need to parse two extra arguments: the minimum and maximum heap sizes. For example, if the minimum heap size is 50 Mb and the maximum is 100 Mb, we type:

java [space] -jar [space] -Xms50m [space] -Xmx100m [space] loracs.jar

Linux/MacOSX command line:

1) Open the Terminal.

2) Move to the directory where the jar file was saved. For example, let’s say the file was saved in the directory /Users/chris/Documents/my\_programs. Then type:

cd [space] /Users/chris/Documents/my\_programs

3) Make sure the jar file is in this directory by typing

ls

4) Start LORACS by typing:

java [space] –jar [space] loracs.jar

If you wish to increase the memory allocated to the Java Virtual Machine (JVM), you’ll need to parse two extra arguments: the minimum and maximum heap sizes. For example, if the minimum heap size is 50 Mb and the maximum is 100 Mb, we type:

java [space] -jar [space] -Xms50m [space] -Xmx100m [space] loracs.jar

**OBS:** Your system administrator can help you determine the amount of heap space that can be allocated to the Java Virtual Machine.

**RUNNING LORACS**

Input files

You will need two input files. Both files are in ascii format and contain a header with 6 lines. You will get an error if the header has a different number of lines. Also the two input files must have the exact same resolution and extent. Look in the directory “example” to see what the files should look like.

1) The **COST** **file** contains, for each pixel, the relative cost to cross this pixel in any direction. Only integers are considered valid (so a value of 2 is OK but 2.0 will cause an error). Also negative values are not considered in the analysis. This means that if you want some pixels to be avoided (for example, water bodies), all you have to do is assign them a negative value. Do not assign a zero value to your pixels as it will probably lead to pathological results.

2) The **ST file** contains the location of the Source/Target patches. Pixels in the Source patch should be marked as -1 and pixels in the Target patch should be marked as -2. All other values will be ignored. All values should be integers (so a value of 2 is OK but 2.0 will cause an error). In general the Source/Target patches are interchangeable but you might get small differences in the MSP results.

This program is intended to compute the paths among patches in a pair wise fashion, so the pixels in the Source patch should be more or less contiguous. The same is true for the Target patch. If you want to obtain results for several pairs of patches you will have to run the program once per pair. When the number of pairs is large it pays to write a bash script or to modify the Java code such that all results are written to a single output file.

Output directory

The outputs will be exactly one directory above the directory you chose in the GUI interface. Unfortunately this is a problem with the Java File Chooser class. Be careful to rename your files if you want to keep copies from multiple runs, because LORACS will overwrite them. There are 3 output files:

1) **CMTC.asc**: this contains the Conditional Minimum Transit Cost values for the pixels located inside the “corridor” (see below: corridor width). Pixels outside the “corridor” are assigned a value of -9999. The cost to move within the Source/Target patches is not included here. That is, the analysis assumes individuals leaving from the outermost pixels in the Source patch and arriving in the outermost pixels in the Target patch. If you want the cost to move within Patches to be considered, then it’s better to select one Source point and one Target point.

Note here that “corridors” are hypothetical structures that result from your choice of movement model (=shortest path), relative cost (=COST file), and corridor width. LORACS is not giving you the location of a wildlife corridor. Rather, you are using LORACS to investigate how your assumptions about habitat suitability and movement behavior translate into landscape connectivity patterns.

2) **MSP.asc**: contains the location of Shortest Paths and the number of times each pixel was included in a Shortest Path. This means if you chose to compute 20 paths, the possible values are 0-20. Larger values indicate movement bottlenecks.

3) **stats.csv**: this is a comma-delimited file that can be read in Excel and contains the cumulative cost and length for each Shortest Path. To get the actual length for each path, multiply the obtained length by the pixel size of your COST file. This list represents an empirical distribution of Shortest Path attributes that can be used to assess significant changes between two runs. For example, one might be interested in computing the impact of removing a conservation unit and run LORACS with and without the conservation unit. A non-parametric “ANOVA” test can be used to compare these two scenarios.

Corridor width

In order to better visualize potential “corridors”, you might want to set a threshold CMTC value above which pixels are not included in the “corridor”. This can be done by adjusting the corridor width. For example, by choosing 20%, only the pixels with the lowest 20% CMTC values are displayed in the results. Values outside the “corridor” are marked as -9999.

Number of Shortest Paths

GIS software (e.g. ArcGIS Spatial Analyst) will give you a single Shortest Path between two points (or two patches). LORACS increases the number of Shortest Paths by making the neighborhood relationships a stochastic function (Pinto and Keitt 2009). The location of Multiple Shortest Paths (MSPs) is saved in the file MSP.asc and their cumulative cost/length is saved in the file stats.csv.

**HOW LARGE CAN MY RASTER BE? HOW MANY PATHS CAN I GET?**

LORACS has been tested on rasters as large as 3\*106pixels. Larger rasters require more heap size to run, and the time to completion increases linearly with the number of shortest paths. The progress bar shows the percentage of shortest paths that has been computed. For a raster with 3\*106pixels, we set –Xms=200m and –Xmx=600m and the time to obtain 100 paths was 187 minutes. This estimate was obtained using a X86 PC laptop with a Pentium dual core processor (2Gb DDR2, 1.73GHz) running Windows Vista 6.0.

As a general rule, if your raster has more than 6 \*104 pixels you will have to start LORACS from the command line. This ensures enough memory is allocated to the Java Virtual Machine. The GUI interface will still work normally – your only interaction with the command line will be writing a single line to start LORACS (see “how do I open the program?” above). Note the above numbers might change if LORACS is not the only process running on your computer.

**REFERENCES**

Pinto N and Keitt TH (2009) Beyond the least-cost path: evaluating corridor redundancy using a graph-theoretic approach. Landscape Ecology 24:253-266.

**SUPPORT**

Help with the program will be provided on a best effort basis. Please write to [naiarapinto@gmail.com](mailto:naiarapinto@gmail.com)

Add the subject “loracs”. In the body of the mail, give a detailed description of the problem and the error message obtained.

**LICENSE**

LORACS is distributed under the terms of the GNU General Public License Version 2, June 1991. The license terms can be found at:

<http://www.gnu.org/licenses/gpl-2.0.html>

This software is free and comes with absolutely no warranty, actual or implied.