Linkage Priority Developer Documentation

Version 2.0—Updated October 2017

John Gallo[1] and Randal Greene[2]
[1]Conservation Biology Institute
[2]Feaver's Lane

Table of Contents

1 Introduction

Linkage Priority (LP) is an ArcGIS 10.x geoprocessing tool within the Linkage Mapper (LM) toolkit. Developers should first understand the LM and LP functionality from the user's perspective. Please review the following user-oriented introductory material:

- Linkage Mapper User Guide version 2.0, including the LM tutorial.
- Linkage Priority User Guide version 2.0, including the LP tutorial. Pay particular attention to the Geoprocessing Overview for high-level introduction to the processing logic that underlies LP.
- Linkage Mapper web site (http://www.circuitscape.org/linkagemapper) and related publications (http://www.circuitscape.org/pubs).

Developers should also be familiar with the ArcGIS geoprocessing framework on which LM and LP are built. Please refer to http://desktop.arcgis.com/en/arcmap/10.4/analyze/main/geoprocessing-framework.htm for an introduction to this framework.

The LM family of tools are built as Python script tools in a standard ArcGIS toolbox. Please refer to http://desktop.arcgis.com/en/arcmap/10.4/analyze/creating-tools/a-quick-tour-of-creating-tools-in-python.htm for an introduction to creating ArcGIS geoprocessing tools in Python.

2 Coding Conventions

As a Python project, developers are encouraged to follow the PEP8 style guide (https://www.python.org/dev/peps/pep-0008/). Please note the following conventions:

- Class, method, function and variable names – lowercase with underscore separators (e.g. `function_name()`).
- Constant "variable" names – all uppercase (e.g. `CONSTANT`).
- Maximum line length – 120 characters. The other LM tools use an 80-character line length, but 120 characters improves readability on the typical modern high-resolution display. We recommend it be adopted for future enhancements to all LM tools.

3 LP Code Organization

Most of the LP functionality is defined in three Python modules (.py files). LP is called by ArcGIS from the Linkage Mapper toolbox, which is defined in two additional files.

ArcGIS is flexible with regards to the installation location of custom toolboxes like Linkage Mapper. However, Linkage Mapper expects all .py files to be in a subdirectory called "scripts".

3.1 lp_settings.py

A module containing "constants" representing advanced LP settings/parameters that are not included in the tool dialog.

3.2 lp_config.py

The LP configuration module, which defines the `lp_config` class and creates an instance of it called `lp_env`. It also has a number of local helper functions, which are described in the LP Classes and Functions section below.

3.3 lp_main.py

The primary LP module, which includes:

- The execution starting point (`if __name__ == "__main__ "`), which calls the `main()` controlling function.
- The `main()` controlling function.
- Functions for the preparation steps.
- The `run_analysis()` function and numerous called functions, which implement the primary analysis logic. The called functions are organized as per the Geoprocessing Overview in the LM user guide, and the Geoprocessing Summary above.
- Local helper functions.
- All functions are described in the LP Classes and Functions section below.

3.4 Summary Diagram of all the modules

- On the following page there is a summary diagram of all the modules in the Linkage Mapper toolbox (including those unrelated to Linkage Priority).
- It gives a good overview of dependancies.
- It is a snapshot in time (2018-04-19).
- Thanks to Darren Kavanagh, using the Pyreverse packacge.

3.5 Linkage Mapper Arc10.tbx

This file defines the Linkage Mapper ArcGIS toolbox. It is modified by right-clicking on it in ArcMap or ArcCatalog and selecting:

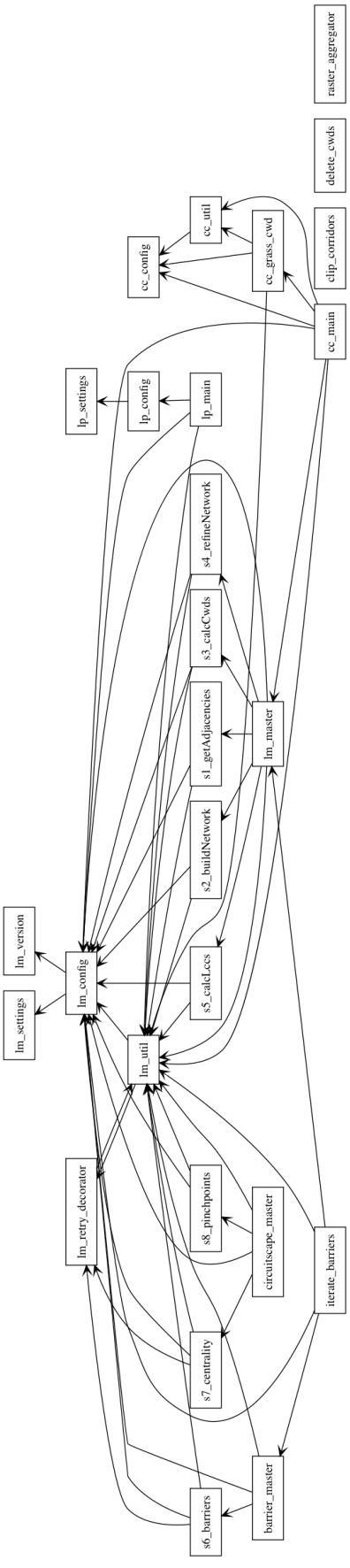
- Item Description, to view the tool metadata (including help text) and edit it.
- Properties, to define the tool's:
    - Name
    - Label
    - Description
    - Optional underlying stylesheet
    - Underlying .py script file
    - Parameters (the type and order of these must match the order in which they are processed in the code)
    - Optional validation code to be run before the .py script is called
    - Optional compiled help file

In general, the toolbox should only be modified when there are new tools, new parameters for existing tools or updated help text.

3.6 LmDlgContent.xsl

This file defines the stylesheet used by the Build Network and Map Linkages tool and the Linkage Priority tool. It facilitates group labeling and nesting of parameters in the tool dialog.

See note on previous page regarding this summary diagram.



4 Geoprocessing Summary

The LP User Guide includes a geoprocessing overview. Additional details are provided here.

- Check that LM in the same Project Directory successfully finished Steps 3 and 5, and terminate if issues
- Make preliminary calculations for each corridor
    - Calculate Permeability
        - In project_LCPs line feature class, calculate the attribute Raw_Perm as LCP_Length / CW_Dist
        - In project_LCPs line feature class, calculate the attribute Rel_Perm as a normalization of all Raw_Perm values
    - Calculate Relative Closeness
        - In project_LCPs line feature class, calculate the attribute Rel_Close as a normalization of all LCP_Length values
    - Invert and normalize
        - For each intermediate corridor raster created by LM, normalize the raster values to create a corresponding intermediate inv_norm raster
- Calculate Core Area Value (CAV) and its components for each core
    - Check weights and warn if issues
    - Add and calculate attributes in the input Core Area Feature Class
        - Mean resistance: mean_res
        - Normalized resistance: norm_res
        - Size: area
        - Normalized size: norm_size
        - Perimeter: perimeter
        - Area/perimeter ratio: ap_ratio
        - Normalized area/perimeter ratio: norm_ratio
        - [Optional] Other core area value: ocav
        - [Optional] Normalized other core area value: nocav
        - [Optional] Expert core area value: ecav
        - [Optional] Normalized expert core area value: necav
        - [Optional] Current Flow Centrality value (from Centrality Mapper): CF_Central
        - [Optional] Normalized CF_Central value: ncfc
        - Core area value: cav
            - (norm_res * Resistance Weight) +
(norm_size * Size Weight) +_
(norm_ratio * Area/Perimeter Weight) +
(necav * Expert Core Area Value Weight) +
(ncfc * Current Flow Centrality Weight) +
(nocav * Other Core Area Value Weight)
        - Normalized core area value: norm_cav
- [Optional] Calculate climate envelope attributes for each core
    - Current climate envelope: clim_env
        - Average of current climate envelope raster values within core

    - Normalized current climate envelope: nclim_env
    - Future climate envelope: fut_clim
        - Average of future climate envelope raster values within core
    - Normalized future climate envelope: nfut_clim
- Complete calculations for each corridor
    - [Optional] Add and calculate attributes of the Core Pairs table
        - Normalize expert corridor importance value: neciv
    - Calculate corridor specific priority (CSP) raster for each corridor
        - Check weights and warn if issues
        - Calculate in-memory variables
            - Average the core area value of the two cores in the corridor: avg_cav
            - [Optional] Difference the climate envelope of the two cores in the corridor: diff_clim_env
                - [Optional] If future climate envelope provided, use future climate envelope for the cooler core
        - Calculate CSP raster:
            - (Rel_Close * Closeness Weight) +
              (Rel_Perm * Permeability Weight) +
              (avg_cav * Core Area Value Weight) +
              (neciv * Expert Corridor Importance Value Weight) +
              (diff_clim_env * Climate Envelope Difference Weight)
        - Apply Proportion CSP Values to Keep setting to reduce size of corridor and create CSP_TOP raster
- Create overall result rasters
    - Combine CSP_TOP rasters using Max and Mean to create overall Corridor Priority Value (project_CPV) raster
        - Calculate sum, count and max of all CSP rasters
        - Calculate project_CPV raster
            - (max * MAXCSPWEIGHT) +
              ((sum / count) * MEANCSPWEIGHT)
    - Clip project_CPV to the MINCPV and normalize to create relative corridor importance (project_RCI) raster
    - Clip project_RCI to extent of truncated raster to create project_linkage_priority raster
    - Invert and normalize truncated raster to create project_NORMTRUNC raster
    - Calculate overall project_blended_priority raster
        - (project_NORMTRUNC * Truncated Corridors Weight) +
          (project_linkage_priority * Linkage Priority Weight)

5 LP Classes and Functions

5.1 lp_config.py

```
def nullfloat(innum):
  """Convert ESRI float or null to Python float."""
  # returns float or None

def nullstring(arg_string):
  """Convert ESRI nullstring to Python null."""
  # returns string or None
```

```
def str2bool(pstr):
  """Convert ESRI boolean string to Python boolean type."""
  # returns Boolean or None

class lp_config(object):
  """Class container to hold Linkage Priority parameters and settings."""
  # instantiated as lp_env

  def configure(self, arg):
  """Assign parameters and settings from passed arguments and advanced settings."""
```

5.2 lp_main.py

```
def delete_datasets_in_workspace():
  """Delete all datasets in workspace."""

def delete_dataset(dataset):
  """Delete one dataset."""

def delete_arcpy_temp_datasets():
  """Delete datasets left behind by arcpy."""

def print_runtime(stime):
  """Print process time when running from script."""

def main(argv=None):
  """Main function for Linkage Priority tool"""

  # preparation steps
  lp_env.configure(argv)
  check_lp_project_dir()
  check_out_sa_license()
  arc_wksp_setup()
  config_lm()
  log_setup()

  # primary analysis
  run_analysis()

  # includes exception handling and shutdown code
  # explicitly or implicitly raised exceptions bubble up to here

def check_lp_project_dir():
  """Checks to make sure path name is not too long. Long path names can cause problems with
  ESRI grids."""

def check_out_sa_license():
  """Check out the ArcGIS Spatial Analyst extension license."""

def arc_wksp_setup():
  """Setup ArcPy workspace."""
```

```
def config_lm():
  """Configure Linkage Mapper."""
  # get log file for last LM run
  # read parameters section from file and turn into tuple for passing
  # use LM config in addition to LP config by passing in parms from last LM run

def log_setup():
  """Set up Linkage Mapper logging."""
  # see LM Architecture below for additional details

def check_add_field(feature_class, field_name, data_type):
  """Check if field exists, and if not then add."""
  # return Boolean indicating whether the field previously existed

def normalize_field(in_table, in_field, out_field, normalization_method,
                    invert=False):
  """Normalize values in in_field into out_field using score range or max score method, with
  optional inversion."""

def normalize_raster(in_raster, normalization_method, invert=False):
  """Normalize values in in_raster using score range or max score method, with optional
  inversion."""
  # return normalized raster

def calc_permeability(lcp_lines):
  """Calculate raw and relative permeability for each Least Cost Path."""

def calc_closeness(lcp_lines):
  """Calculate relative closeness for each Least Cost Path."""

def inv_norm():
  """Invert and normalize each corridor."""
  # could be multiple nlc folders

def cav():
  """Calculate Core Area Value (CAV) and its components for each core."""
  # check weights and warn if issues
  # check/add fields
  # calc mean resistance
  # calc area, perimeter and ratio
  # normalize CAV inputs
    # resistance – invert
    # size
    # area/perimeter ratio
    # optionally ecav
    # optionally cfc
  # optionally calc OCAV
  # calc CAV
  # normalize CAV with score range normalization

def clim_env():
```

```
    """Calculate Climate Envelope(s) for each core."""
    # calc score range normalization on current climate envelope
    # calc aerial mean climate envelope for each core
    # score range normalize resulting values
    # optionally repeat for future climate envelope

def eciv():
    """Normalize Expert Corridor Importance Value (ECIV) for each corridor."""

def csp(sum_rasters, count_non_null_cells_rasters, max_rasters, lcp_lines):
    """Calculate Corridor Specific Priority (CSP) for each corridor."""
    # check weights
    # could be multiple folders
        # process each corridor raster in folder
            # check for max 1
            # get cores from raster name
            # check for corresponding link
            # get and avg CAVs for the core pair
            # optionally get ECIV for the core pair
            # optionally get and difference climate envelopes for the core pair to create CED
                # optionally use future climate envelope for cooler core
            # calc weighted sum CSP raster
            # get max and min
            # determine and apply threshold from Proportion Top CSP Values to Keep
    # perform intermediate calculations on CSPs leading toward CPV

def cpv(sum_rasters, count_non_null_cells_rasters, max_rasters, cpv_raster):
    """Combine CSPs using Max and Mean to create overall Corridor Priority Value (CPV)."""

def rci(cpv_raster, rci_raster):
    """Clip CPV to the MINCPV and renormalize to create relative corridor importance (RCI)."""

def linkage_priority(rci_raster, trunc_raster, lp_raster):
    """Clip RCI to extent of truncated raster (LP)."""

def norm_trunc(trunc_raster, norm_trunc_raster):
    """Invert and normalize truncated raster (NORMTRUNC)."""

def blended_priority(norm_trunc_raster, lp_raster, bp_raster):
    """Calculate overall Blended Priority."""

def run_analysis():
    """Run main Linkage Priority analysis."""
    # check that LM finished with steps 3 and 5
    # check/create gdb for scratch
    # check/create gdb for intermediate
    # set key dataset locations
    # calc permeability
    # calc relative closeness
    # invert and normalize each corridor
```

```
# calc Core Area Value (CAV) and its components for each core
# normalize Expert Corridor Importance Value (ECIV)
# calc climate envelope
# calc Corridor Specific Priority (CSP)
# calc Corridor Priority Value (CPV)
# calc Relative Corridor Importance (RCI)
# calc Linkage Priority (LP)
# calc Blended Priority (BP)
# save a copy of Cores as the "Output for ModelBuilder Precondition"
```

6 LM Architecture

As detailed above, LP modifies and builds on the standard LM outputs using the storage structure established by LM. As part of the LM family of tools, LP also takes advantage of some shared elements of the LM architecture.

6.1 Configuration and Settings

For accessing the details of the successful LM run that LP builds on, LP instantiates and fills the `lm_config()` class (defined in lm_config.py) as `lm_env`. This includes access to LM's advanced settings defined in lm_settings.py.

6.2 Logging

LM provides logging to both the ArcGIS geoprocessing framework (for display and geoprocessing results history within ArcGIS) and to text files within the LM project structure (in the run_history folder). LP uses the following functions logging functions from lm_util.py:

```
def create_log_file(messageDir, toolName, inParameters):
  """creates and new text file for logging and remembers it for the current run"""

def write_log(string):
  """write the string to the current log file"""

def close_log_file():
  """write the current time to the current log file and close the file"""

def gprint(string):
  """write the string to the current log file and pass to the ArcGIS geoprocessing framework as
  a message or warning"""

def raise_error(msg):
  """write the message to the current log file, pass it to the ArcGIS geoprocessing framework
  as an error, and close the log file"""
```

6.3 Other Utilities

LP also uses the following functions from lm_util.py:

```
def build_stats(raster):
  """Builds statistics and pyramids for output rasters"""
```

```
def delete_data(dataset):
  """Delete the passed ArcGIS dataset"""
```

7 Other Notes

7.1 Code Editing and Debugging

The default Python editor is fairly rudimentary and, for many developers, not appropriate for larger projects like the LM family of tools. Code-aware text editors like notepad++ are an improvement, but do not offer interactive debugging. For interactive debugging, a Python-aware integrated development environment (IDE) is recommended. We have successfully used:

- PyCharm (https://www.jetbrains.com/pycharm/)
- Microsoft Visual Studio Community Edition (https://www.visualstudio.com/vs/community/)

Unfortunately, the current ArcGIS architecture does not support interactive debugging of Python geoprocessing tools when the tools are run from within ArcGIS. However, the code has been setup to support interactive debugging by starting the tool from within an IDE. Do this by:

- Opening lp_main.py and going to the execution starting point (`if __name__ == "__main__"`).
- Commenting out the call to `main()`.
- Uncommenting out the subsequent block that builds an `args` list and passes it by calling `main(args)`.
- Changing `args` to contain values relevant for the scenario to be debugged.
- Indicating to your IDE that lp_main.py is the startup module.
- Starting the program in debug mode.
- Using breakpoints, watches and other interactive debugging techniques.

7.2 Source Control Using GitHub

The LM family of tools are managed as an open source repository on GitHub. The repository URL is https://github.com/linkagescape/linkage-mapper.

7.3 Becoming a Linkage Mapper Contributor

We encourage contributions to the LM project by ArcGIS/Python developers. This could include enhancements and fixes to existing tools, and development of new tools for the LM toolbox. We encourage new tools to follow the protocols in Linkage Priority and Climate Linkage Mapper, which are currently the two newest tools in the LM toolbox.

7.4 Project Management Using Trello

Depends on outcome of discussion with Tim and Darren.
Does this belong here, or is it just for CBI staff/contractors?

7.5 Future Enhancements

As with any active software project, there have been numerous suggestions for future enhancements. These include:

- Adding normalization method options (max score vs. score range) everywhere LP does normalization.
- Adding additional details to the LP log file, such as the parameters passed from ArcGIS (currently, these are accessible only from the ArcGIS geoprocessing results history).
- Improving tutorial with additional examples demonstrating LP's optional parameters.

Please use the LM User Group (see the Support section of the LM User Guide) to register your suggestions.