Circuitscape: A Tool for Landscape Ecology

Viral B. Shah (vshah@interactivesupercomputing.com) – *Interactive Supercomputing, Waltham, MA.* USA
Brad McRae (bmcrae@tnc.org) – *The Nature Conservancy, Seattle, WA.* USA

The modeling of ecological connectivity across networks and landscapes is an active research area that spans the disciplines of ecology, conservation, and population genetics. Recently, concepts and algorithms from electrical circuit theory have been adapted to address these problems. The approach is based on linkages between circuit and random walk theories, and has several advantages over previous analytic approaches, including incorporation of multiple dispersal pathways into analyses. Here we describe Circuitscape, a computational tool developed for modeling landscape connectivity using circuit theory. Our Python implementation can quickly solve networks with millions of nodes, or landscapes with millions of raster cells.

Introduction

Modeling of ecological connectivity across landscapes is important for understanding a wide range of ecological processes, and for achieving environmental management goals such as conserving threatened plant and animal populations, predicting infectious disease spread, and maintaining biodiversity [Cro06]. Understanding broad-scale ecological processes that depend on connectivity, and incorporating connectivity into conservation planning efforts, requires quantifying how connectivity is affected by environmental features. Thus, there is a need for efficient and reliable tools that relate landscape composition and pattern to connectivity for ecological processes.

Recently, concepts and algorithms from electrical circuit theory have been adapted for these purposes ([Mcr06], [Mcr08]). The application of circuit theory to ecological problems is motivated in part by intuitive connections between ecological and electrical connectivity: as multiple or wider conductors connecting two electrical nodes allow greater current flow than would a single, narrow conductor, multiple or wider habitat swaths connecting populations or habitats allow greater movement between them. In addition, rigorous connections between circuit and random walk theories [Doy84] mean that current, voltage, and resistance in electrical circuits all have concrete interpretations in terms of individual movement probabilities [Mcr08]. Such models can be useful for conservation planning and for predicting ecological and genetic effects of spatial heterogeneity and landscape change; for example, effective resistances calculated across landscapes have been shown to markedly improve predictions of gene flow for plant and animal species [Mcr07].

Here we describe Circuitscape, a computational tool which applies circuit-theoretic connectivity analyses to landscape data using large-scale combinatorial and numerical algorithms [Sha07].[1]

Applying circuit theory to predict landscape connectivity

In spatial ecology and conservation applications, landscapes are typically mapped as grids of raster cells in a geographical information system (GIS). For connectivity analyses, grid cells represent varying qualities of habitat, dispersal routes, or movement barriers. These raster grids can be represented as graphs, with each grid cell replaced by a node and connected to its neighbors by edges, with edge weights proportional to movement probabilities or numbers of migrants exchanged. Edges are assumed to be undirected, which implies that dispersal is balanced. Heterogeneity in landscape characteristics will typically cause movement probabilities to vary, resulting in graphs with heterogeneous edge weights.

These graphs can be analyzed using circuit theory to predict different aspects of connectivity and movement probabilities. Overviews of the theory and applications of circuit theory in ecology, conservation, and genetics are presented in [Mcr06] and [Mcr08], and will only be summarized here. Briefly, effective resistance across networks can be used as both a distance measure and a measure of redundancy in connections across graphs, and can be related to random walk times between nodes [Cha97]. Current flowing across a graph also has interpretations in terms of random walks, with current densities along circuit branches reflecting net passage probabilities for random walkers passing through nodes or across edges [Doy84]. Similarly, voltages measured in circuits can be used to predict probabilities that a random walker will reach one destination (e.g., a habitat patch) or state (e.g., death) before another [Doy84].

Computing resistance, current, and voltage with Circuitscape

Circuitscape was developed to apply circuit theory to problems in landscape ecology, which can require operations on large graphs. The computation typically starts with the raster cell map of a landscape exported from a GIS. The landscape is coded with resistance or conductance values assigned to each cell based on landscape features, such that conductance values are proportional to the relative probability of movement through each habitat type. Circuitscape converts the landscape into a graph, with every cell in the landscape represented by a node in the graph. Thus, an *mn* cell map results in a graph with $k = mn$ nodes. Connections between neighboring cells in the landscape are represented as edges in the graph. Typically, a cell is connected to either its 4 first-order neighbors or its 8 first and second-order neighbors, although long distance connections are possible. Edge weights in the graph are functions of the per-cell conductance values, usually either the average resistance or average conductance of the two cells being connected. More sophisticated ways of computing edge weights may also be used.

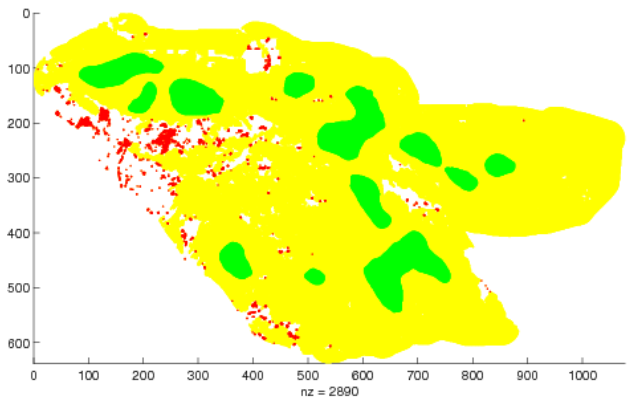
In the simplest case, we are interested in computing effective resistances, voltages, and current densities between pairs of nodes on a graph. This is done with Kirchoff's laws in matrix form. Let g_{ij} denote the conductance of the resistor connecting nodes i and j. Let G be an $n \times n$ weighted Laplacian matrix, such that $G_{ij} = -g_{ij}$ and $G_{ii} = \sum_{j=1}^{n} g_{ij}$. Resistance between nodes x and y (with $x < y$ for convenience) may be computed using a reduced conductance matrix G_y, which is the same as G but with the y^{th} row and column removed. The right hand side I is a vector with all zeros except in the x^{th} position where it is set to one. Now, solving $G_y v = I$ yields the vector v, which can then be used to derive effective resistances between nodes x and y, as well as current densities across the graph. Multiple current or voltage sources, multiple grounds, and connections to ground via resistors can be accomodated with minor modifications to this method.

The size of the graph depends on both the extent of the landscape and the resolution of the landscape data. The area modeled can vary widely across different ecological studies or conservation efforts; extents of ecological studies may be as small as a few square meters, or as large as a continent. Conservation efforts may focus on a single property, or be as large as the Yellowstone-to-Yukon project, which extends for more than 2000 miles from Yellowstone National Park to the Yukon's Mackenzie Mountains. The appropriate resolution, or cell size, for analyses will depend on kind of animal being modeled. The amount of land an animal perceives around itself typically depends on its size and tendency to move within its environment: a mountain lion may perceive about 10,000 square meters of land around it, whereas a mouse may only perceive a few square meters. Applying circuit theory to model connectivity requires working with a resolution fine enough to match the species being modeled, and an extent that may fall anywhere in the range described above. As a result, graph sizes get large very quickly. For example, a 100 km^2 area modeled for mountain lions with 100m cell sizes would yield a graph with 10,000 nodes. A landscape that includes the entire state of California would result in a graph with 40 million nodes. Landscapes that span several states can easily result in graphs with hundreds of millions of nodes; the Yellowstone-to-Yukon region includes 1.2 million km^2 of wildlife habitat, requiring 120 million nodes at 100m resolution.

Computational Methods

Circuitscape performs a series of combinatorial and numerical operations to compute a resistance-based connectivity metric. The combinatorial phase preprocesses the landscape for the subsequent numerical operations that compute resistance, current, and voltage across large graphs.

Combinatorial Methods



Combinatorial preprocessing of a landscape.

Circuitscape first reads the raster cell map from a file and constructs a graph. The raster cell map is represented by an $m \times n$ conductance matrix, where each nonzero element represents a cell of interest in the landscape. Each cell is represented by a node in the graph. Given a node in the graph, the graph construction process inserts an undirected edge connecting the node with its 4 or 8 neighbors. As a result, the graph has up to 5 or 9 nonzeros per row/column, including the diagonal. The choice of neighbor connectivity can affect connectivity of the resulting graph. A graph that is connected with 8 neighbors per cell may not be connected with 4 neighbors per cell.

The landscape graph is stored as a sparse matrix of size $mn \times mn$. This graph is extremely sparse due to the fact that every cell has at most 4 or 8 neighbors, even though the landscape may be extremely large. Edges in the graph are discovered with stencil operations. Once all the edges are discovered, the graph is converted from the triple (or co-ordinate) representation to an efficient representation such as compressed sparse rows.

A habitat patch in a landscape is represented as a collection of several neighboring nodes in the graph. Since we are typically interested in analyzing connectivity between two or more habitat patches (e.g., the patches shown in green in the figure above), all nodes in a focal habitat patch are considered collectively, and contracted into one node. Neighbors of the nodes of a habitat patch are now neighbors of the contracted node; this introduces denser rows and columns in the sparse matrix, or higher degree nodes in our graph.

Finally, we need to ensure that the graph is fully connected. Physically, there is no point in computing current flow across disconnected pieces of the graph. Numerically, it leads to a singular system. Circuitscape ensures that the source and destination habitat patches are in the same component, and it can iterate over source destination pairs in disjoint pieces of the landscape. We prune the disconnected parts of the landscape by running a connected components algorithm on the landscape graph. In the example above, the nodes shown in red are pruned when computing current flow between the green habitat patches.

We note that the combinatorial preprocessing is not performed just once, and may need to be performed for each source/destination pair. Thus, it has to be quick even for extremely large graphs. We touch upon the software and scaling issues in a later section.

Numerical Methods

Once the graph is constructed, we form the graph Laplacian; this simply consists of making all off-diagonal entries negative and adjusting the diagonal to make the row and column sums zero. A row and column are then deleted from the graph Laplacian (making the matrix symmetric positive definite) corresponding to the destination node, indicating that it is grounded.

Effective resistance, current flows and voltages can then be computed by solving a linear system. Rows and columns corresponding to nodes connected directly to ground are deleted from the graph Laplacian, and diagonal elements corresponding to nodes connected to ground by resistors are altered by adding the conductance of each ground resistor. These modifications to the Laplacian make the matrix symmetric positive definite. The right hand side of the system is a vector of all zeros except in the position of source nodes, which are given values corresponding to the amount of current injected into each.

For small to moderate problem sizes, direct methods work well. In cases with one destination and multiple sources, the Cholesky factorization can be computed once, and then solutions can be achieved with triangular solves.

Optimal Cholesky decomposition of the 2D model problem on a unit square requires $O(n \log n)$ space and $O(n^{3/2})$ time. Although the space requirements seem modest asymptotically, they are prohibitive in practice for large problems, as shown in the table below. The number of floating point operations for the largest problems is also prohibitive. The analysis for the 2D model problem holds for matrices generated from landscapes represented as 2D grids.

Cells (10^6)	Fill (10^6)	GigaFlops
0.25	6.3	0.61
1	30	5.5
12	390	200
48	1800	1700

Time and space requirements to use sparse direct solvers.

We chose to explore iterative methods due to the large amount of memory required for Cholesky factorization, coupled with the amount of time it would take on large grids. We use preconditioned conjugate gradient to solve the linear systems with an algebraic multigrid preconditioner [Sha07].

A synthetic problem

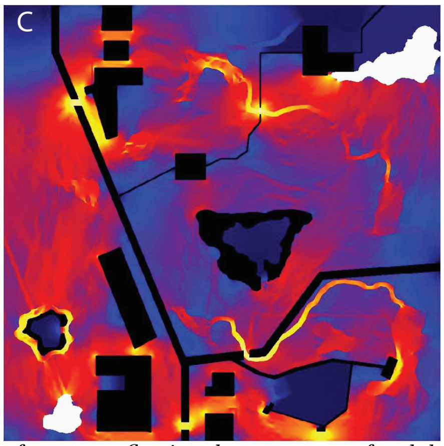
We provide some preliminary performance of Circuitscape on a synthetic problem shown below. The larger problem sizes are tiled versions of the smaller problem. While this does not represent a real application scenario, it does test the scaling of our algorithms and code on large problem sizes.

The synthetic landscape used for performance testing [Mcr08].

The largest problem we could solve had 6 million nodes. On a problem twice as large (12 million), we notice memory allocation failure (even though we believe we have sufficient memory). In the table below, we report times for the different phases of the problem.

Size	Build Graph	Components	AMG setup	Linear solve
1M	3 sec	5 sec	4 sec	9 sec
6M	14 sec	27 sec	31 sec	82 sec

Performance results for the synthetic landscape.



Map of current flowing between two focal habitat patches. Current densities through cells indicate the probability of a random walker passing each cell as it moves from one patch to the other. The map highlights portions of the landscape critical for movement between the focal patches [Mcr08].

Implementation in Python

Circuitscape was first implemented in Matlab. We decided to move to Python primarily for flexible scripting, platform independence, and potential for integration with ArcGIS (ESRI, Redlands, California, USA). The Python implementation of Circuitscape has its own GUI (designed with PythonCard), through which the user specifies the inputs and chooses from the several different problem types available.
We use numpy to manage our dense arrays, but use the sparse matrix functionality from scipy to store and manipulate the landscape graph. We use the coordinate and compressed sparse row (CSR) storage formats from scipy . We also use the conjugate gradient solver from scipy along with the algebraic multigrid preconditioner from pyamg.
One of the shortcomings of various array-based computing tools is the lack of support for operations that work with other data structures. We searched for libraries that would let us perform graph operations at scale, and were unable to find one that suited our needs. We instead wrote our own graph processing module, implementing many graph algorithms with array primitives [Sha07].

Looking forward

We are constantly trying to push the envelope on the largest problem we can solve. We are experimenting with Circuitscape on computers ranging from desktops to multiprocessor shared memory systems with 64G of RAM. We can solve problems with 6 million nodes, but larger problems appear to be restricted by memory. We are currently investigating tools that can help us identify and reduce the memory footprint of our application.

We are also working on parallelizing Circuitscape. The Matlab version of Circuitscape parallelized effortlessly with Star-P (Interactive Supercomputing, Waltham, Massachussetts, USA). For the Python implementation of Circuitscape, we are taking a different approach to parallelization. We will initially start with task-parallelism to solve several moderate sized problems simultaneously. We also plan to parallelize the linear solver, allowing users to perform connectivity computations across extremely large landscapes. Finally, we hope to integrate Circuitscape with ArcGIS to make it easy for users of ArcGIS to perform circuit-based connectivity analyses.

Circuitscape will be released under an open source license.

References

[Cha97] Chandra, A. K., P. Raghavan, W. L. Ruzzo, R. Smolensky, and P. Tiwari. 1997. The electrical resistance of a graph captures its commute and cover times. Computational Complexity 6:312–340.

[Cro06] Crooks, K. R. and Sanjayan M. (eds). 2006. Connectivity Conservation. Cambridge University Press.

[Doy84] Doyle P. G. and Snell, J. L. Random walks and electrical networks. 1984. Mathematical Association of America.

[Mcr06] McRae, B. H. 2006. Isolation by Resistance. Evolution 60:1551-1561.

[Mcr07] McRae, B. H. and Beier, P. 2007. Circuit theory predicts gene flow in plant and animal populations. Proceedings of the National Academy of Sciences of the USA 104:19885-19890.

[Mcr08] McRae, B. H., Dickson, B. G., Keitt, T. H., and Shah, V. B. In press. Using circuit theory to model connectivity in ecology and conservation. Ecology.

[Sha07] Shah, V. B. 2007. An Interactive System for Combinatorial Scientific Computing with an Emphasis on Programmer Productivity. PhD thesis, University of California, Santa Barbara.

[1]The Wilburforce Foundation sponsored this work. B. McRae is supported as a Postdoctoral Associate at the National Center for Ecological Analysis and Synthesis, a Center funded by NSF (Grant #DEB-0553768), the University of California Santa Barbara, and the State of California.