[Re] Least-cost modelling on irregular landscape graphs

Joseph Stachelek[1]

1 South Florida Water Management District, West Palm Beach, Florida, USA

jstachel@sfwmd.gov

Editor
Name Surname

Reviewers
Name Surname
Name Surname

Received Sep, 1, 2015
Accepted Sep, 1, 2015
Published Sep, 1, 2015

Competing Interests:
The authors have declared that no competing interests exist.

A reference implementation of

→ Least-cost modelling on irregular landscape graphs, T. Etherington, Landscape Ecology, 2012.

Introduction

We propose a reference implementation of [3] that describes a method for generating accumulated cost surfaces using irregular landscape graphs. Accumulated cost surfaces are commonly used in landscape ecology, autonomous navigation, and civil engineering to represent travel costs and connectivity among points in a spatial domain. The constuction of these surfaces depends on an underlying landscape graph made up of nodes and distance-weighted edges. Conventionally, landscape graphs are constructed from a complete set of all possible nodes in the domain. The original article explored the use and constuction of irregular landscape graphs which only accounts for an "intelligent" subset of all possible nodes.

According to the original article, irregular landscape graphs allow for faster processing speeds and avoid directional bias relative to regular landscape graphs. The original implementation was made in Python whose sources are available upon request to the author of the original article. The reference implementation we propose has been coded in R because of the strength of existing libraries for generating accumulated cost surfaces using regular landscape graphs [4].

Methods

We used the description of the model as well as the source code of the original implementation (requested from the author) as the basis for the following reference implementation. We attempted to follow the structure, style, and order-of-operations of the original with a few exceptions. For example, we use the same underlying Fortran algorithm for computing the Delaunay triangulations [1] that form the basis of irregular landscape graph construction. One notable difference in our reference implementation relative to the original is that we have used matrix operations from the gdistance package [4] rather than nested loops to contruct our regular landscape graphs.

Results

First, we reproduced the basic output of Figures 3 and 4 using model inputs obtained from the author of the original article. We tested a range of different algorithms for

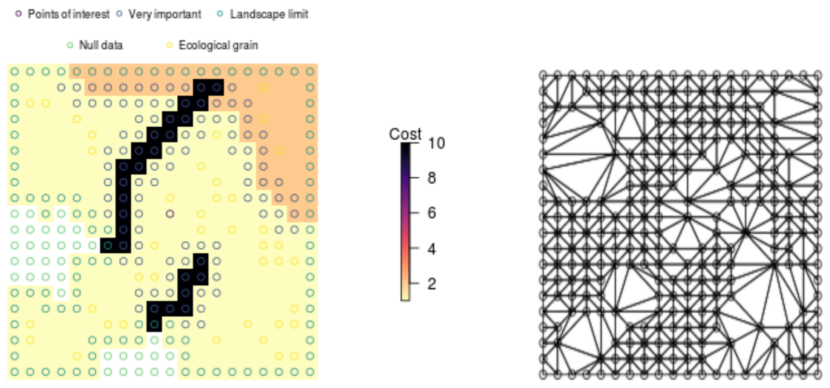

Figure 1: Node-edge selection ensures that all relevant landscape features are retained in the accumulated cost surface. Note that the triangulation in the second panel incudes Null data nodes. These are trimmed prior to construction of the final graph.

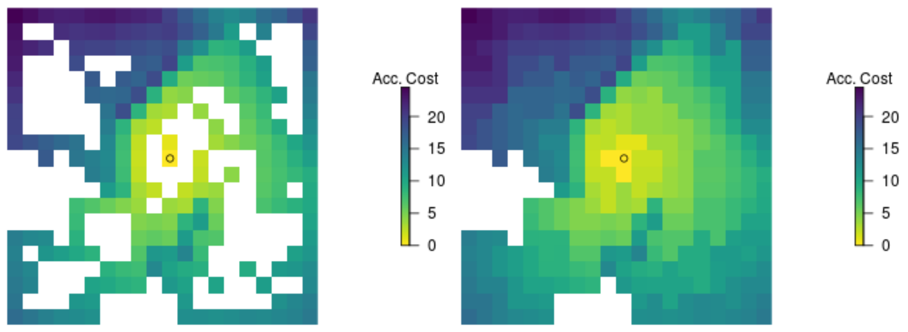

Figure 2: Accumlated cost surface construction begins by traversing the graph from the starting-node (open circle) to the remaining points in the landscape graph. In the final step, missing nodes are imputed according to a nearest neighbor selection.

producing Delaunay triangulations before settling on the same underlying algorithm as the original [1].

Next, we reproduced the performance comparisons in Figure 7. Our findings suggest a more nuanced interpretation of the relative performance of the two methods. Although initial construction was much faster for regular landscape graphs, at a sufficiently high number of starting nodes the initial performance penalty afforded to irregular landscape graphs was outweighed by the decrease in per starting-node processing time. We attribute these findings to the fact that the simple structure of regular landscape graphs is amenable to matrix operations and that irregular landscape graphs have a lower number node/edge features.

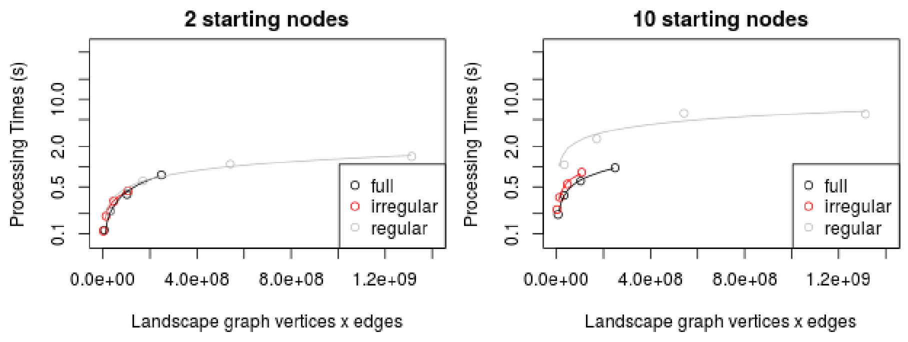

Figure 3: Performance comparisons between regular and irregular landscape graphs. Note that for only two source cells, the performance benefit realized by the irregular landscape graph was outweighed by a higher initialization cost.

Profiling of the reference implementation code revealed that the bulk of the processing time required to construct irregular landscape graphs was spent on Delaunay triangulation. Note that our reference implementation uses compiled Fortran code [1] to implement Delaunay triangulations and compiled C code from the igraph package [2] to construct graphs and calculate accumulated cost distances.

Finally, we reproduced the directional bias tests in Figure 8. As in the original article, we found that regular graphs produced directionally-biased cost surfaces. However, we were able to correct for these biases using the `gdistance::geoCorrection` function. This correction involves scaling graph edge weights by the diagonal distance between grid cells.

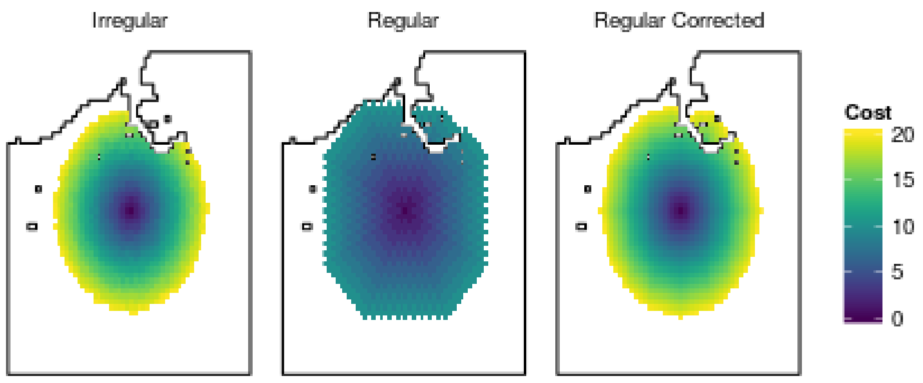

Figure 4: Comparison of directional bias between irregular, regular, and corrected-regular landscape graphs

Conclusion

We were able to replicate the finding of the original article that irregular landscape graphs provide a performance benefit relative to regular landscape graphs but this was true only under certain conditions. We found that although irregular landscape graphs suffer a high initialization cost relative to regular landscape graphs they have a lower individual (per-unit) starting-node processing time. Potential users of irregular landscape graphs should consider initialization and performance trade-offs prior to implementation.

References

[1] C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa. "The quickhull algorithm for convex hulls". In: *ACM Transactions on Mathematical Software (TOMS)* 22.4 (1996), pp. 469–483.

[2] Gabor Csardi and Tamas Nepusz. "The igraph software package for complex network research". In: *InterJournal* Complex Systems (2006), p. 1695. URL: http://igraph.org.

[3] Thomas R Etherington. "Least-cost modelling on irregular landscape graphs". In: *Landscape ecology* 27.7 (2012), pp. 957–968.

[4] Jacob van Etten. *gdistance: Distances and Routes on Geographical Grids*. R package version 1.1-9. 2015. URL: http://CRAN.R-project.org/package=gdistance.