On the Use of Fast Marching Algorithms for Shortest Path Distance Calculation

J.B. Boisvert

There are many uses to shortest path algorithms; in past CCG papers the shortest path algorithm has been used to incorporate locally varying anisotropy into geostatistical modeling. The Dijkstra algorithm has been analyzed extensively in past CCG papers; this paper presents an alternative methodology based on fast marching methods. The methodology is not restricted to a graph reformulation as the Dijkstra algorithm is, thus, the memory requirements of the algorithm are smaller while the CPU time required is typically larger. The two shortest path algorithms are analyzed and compared; it is found that the fast marching algorithm provides similar distance calculations as the Dijkstra algorithm with considerably less RAM requirements.

1. Introduction

The Dijkstra algorithm has been used in a methodology for incorporating locally varying anisotropy into geostatistical modeling (Boisvert, 2010). An integral step in the LVA methodology is the calculation of the shortest path distance between locations. Two algorithms are compared for the calculation of the shortest path distance (1) the Dijkstra algorithm and (2) fast marching methods (FMM).

2. Fast marching method

The fast marching algorithm was introduced by Sethian (1999). Essentially, it is a numerical algorithm to solve for the viscosity solution of the Eikonal equation (Equation 1). Conceptually, consider an advancing front with some speed F. The distance to any location (from some starting node) is analogous to the time required for the advancing front to reach that location.

$$\left|\nabla T\right| F = 1 \tag{1}$$

where *F* is the speed perpendicular to the advancing wave front and *T* is the arrival time.

The implementation utilized is a C++ Matlab integrated library. The source is freely available from Peyre (2008). The only inputs required are the LVA field and the grid. The stepsize taken for each iteration of the fast marching algorithm is equal to the grid cell size; thus, the accuracy of the distances can be increased by decreasing the cell size of the grid; the accuracy of the distances from the Dijkstra algorithm are controlled by the number of offsets connecting cells in the graph, see Boisvert(2010) for more details.

3. Comparison between Dijkstra and Fast Marching

There are three areas of interest when comparing the shortest path algorithms (1) RAM memory usage (2) Run time and (3) accuracy of the resulting distances. These three criteria are explored using a series of synthetic data sets. For the purposes of shortest path calculations, the only requirement is an LVA field, the data are not relevant. Consider two LVA fields that are used to demonstrate the advantages and disadvantages of the two algorithms, the first LVA field is a sinusoidal wave (in x and y) with a constant anisotropy ratio of 20:1 (Figure 1, data provided by Maptek). The second LVA field is a constant dip of 10˚ East that will be used to demonstrate the limitations of considering a graph (Dijkstra).

4. Run Time Comparison

First, consider determining the shortest path in the sinusoidal LVA field. For the proposed methodology (Boisvert 2010) the shortest path between a single landmark point and all locations in the grid is required. The CPU time required to calculate the necessary distances with the fast marching algorithm depends on the number of blocks in the grid. The CPU time required to calculate the necessary distances with the Dijkstra algorithm depends on the

number of blocks in the grid as well as the number of offsets selected (Figure 2) to define the graph; with the Dijkstra algorithm, paths are limited to edges, edges are built between nearby vertices (grid cells), more offsets results in more accurate paths but requires more CPU time and RAM, see Boisvert (2010) for details.

The run times presented in Figure 3 for the fast marching and Dijkstra algorithms were generated on an Intel(R) Xeon(R) X5677@3.47GHz computer running 64-bit Windows 7. From these times, it is clear that the Dijkstra algorithm is slightly faster than the fast marching algorithm when considering a few offsets; however, increasing the number of offsets beyond 2 results in slower times due to the large number of edges in the graph.

While the Dijkstra algorithm is slightly faster than the fast marching algorithm (Figure 3) it uses a significantly larger amount of RAM (Figure 4). It should be noted that both the CPU speed and RAM requirements of these algorithms are dependent on the details particular to the implementation of each algorithm. Different implementations will have different requirements; however, the Boost Graphical Library is a well used implementation of the Dijkstra algorithm and there are few publically available alternatives to Sethian (1999) for FMM algorithms that implement LVA.

While the 40% increase in CPU time required for the fast marching algorithm may be significant for large grids with many landmark points where the algorithm must be run multiple times (+100), the 40% increase is likely not a deterrent for using fast marching given the significantly reduced RAM requirements. Often the RAM requirements of the Dijkstra algorithm will be too high for a typical personal computer and FMM is an attractive alternative.

The final comparison between the methodologies is based on the accuracy of the paths. The main issue with the Dijkstra algorithm is that it generates paths that are restricted to the user defined edges in the graph (Figure 5). This can have significant consequences if the LVA is in a direction that is not closely aligned with the orientation of the edges. Consider a constant LVA field with a dip of 22.5˚. Using the Dijkstra algorithm with a single offset the edges are restricted to 45˚ increments and cannot accurately capture the anisotropy (Figure 6, top right). The fast marching algorithm has little difficult with this LVA field as it is continuous and not restricted to edges in predefined orientations. Increasing the number of offsets in the Dijkstra algorithm to two removes the artifact caused by the misalignment of the anisotropy with the edges in the graph.

In order to obtain accurate paths the Dijkstra algorithm can be implemented with additional offsets; however, this may not be possible for large grids as the RAM requirements increase dramatically with even 2 offsets (Figure 4). It has been the authors' experience that using 2 offsets is nearly always sufficient to remove the artifacts generated with the Dijkstra algorithm.

The 22.5˚ dipping LVA field was the worst case scenario for the potential distance errors with the Dijkstra algorithm. When the sinusoidal LVA field is used, even a single offset with the Dijkstra algorithm is sufficient to calculate reasonable distances (Figure 7). In general, the Dijkstra algorithm with more offsets results in better (shorter) distances (Figure 8); however, fast marching is slightly better for the larger distances. Note that implementing the Dijkstra algorithm with offsets larger than 2 is rarely possible given practical grid sizes.

Fast marching offers a graphless alternative to the Dijkstra algorithm. The implementation is still restricted to a regular grid but distances along any direction can be incorporated. The distances can be improved by decreasing the cell size which decreases the step size used when solving for the distances; however, this has little effect on the overall solution (Figure 6).

Table 1: Comparison of algorithms.

Algorithm	RAM Usage	Run Times	Benefits	Drawbacks
Dijkstra	Very high	Moderate	-Run times are faster -accurate distances if a sufficient number of offsets are used	-difficult to select number of offsets
Fast Marching	High	Slow	-no need to select number of offsets -flexible distance calculation (can consider shallow dips)	-little ability to refine distances to increase accuracy -More CPU intensive than Dijkstra, although not enough to discount fast marching (similar speed if using 2 offsets with Dijkstra)

5. Conclusions

Two algorithms for the calculation of the shortest anisotropic distance between locations in the presence of LVA were presented. Both algorithms have strengths and weaknesses; regardless, some recommendations are required to assist users in selecting the appropriate algorithm. The Dijkstra algorithm is slightly faster and in general seems to return slightly shorter (more accurate) paths if a sufficient number of offsets (>1) are selected. The user is recommended to implement the Dijkstra algorithm if there are sufficient CPU resources to consider 2+ offsets, otherwise FMM is preferred to avoid artifacts due to the restrictions on the paths in the Dijkstra algorithm.

References

Boisvert J, 2010. Geostatistics with Locally Varying Anisotropy. PhD Thesis. University of Alberta. 175 p.

Peyre G, 2008. Toolbox Fast Marching - A toolbox for Fast Marching and level sets computations. http://www.mathworks.com/matlabcentral/fileexchange/6110. Accessed April 2010.

Sethian, 1999. Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science. Cambridge University Press. 404p.

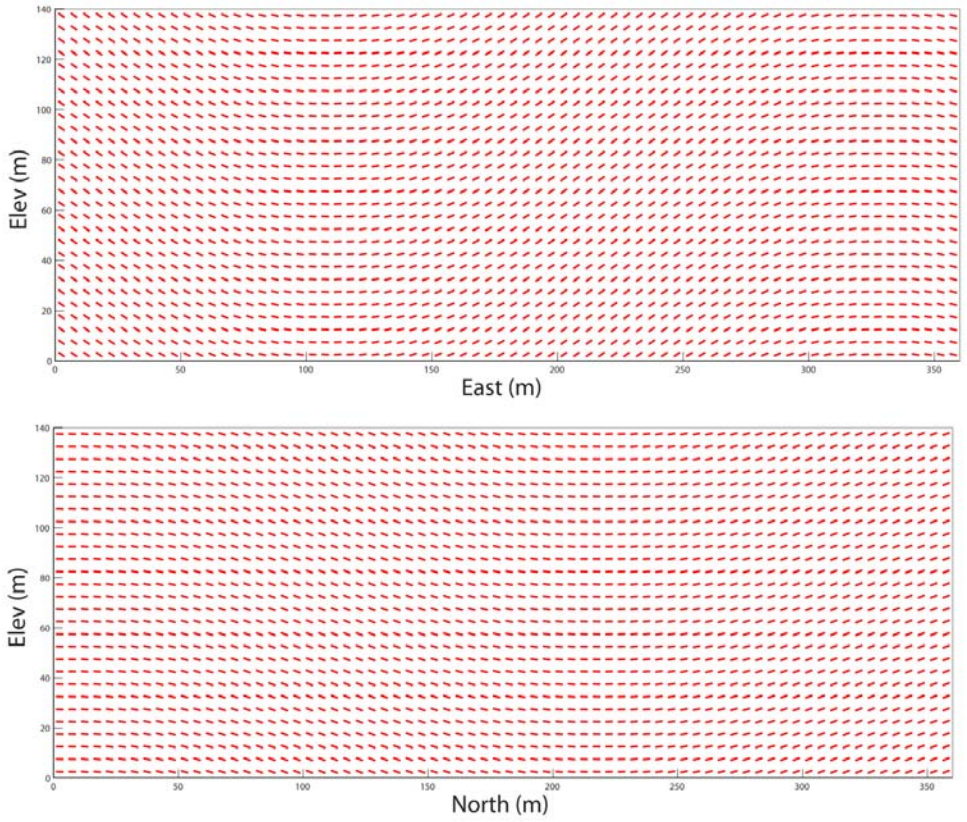

Figure 1: Sinusoidal LVA field. A constant anisotropy ratio of 20:1 is used.

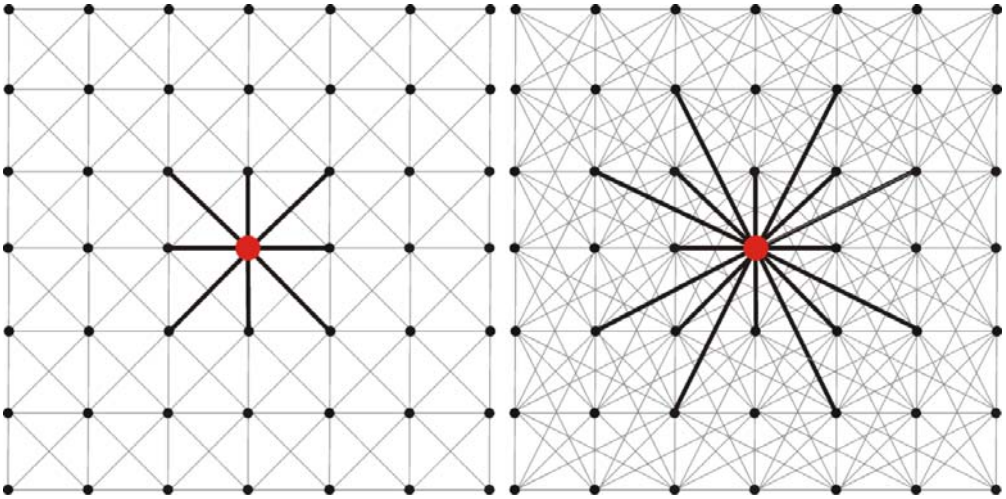

Figure 2: Left: Graph with vertices connected by 1 offset. Right: Graph with vertices connected by 2 offsets. Thicker lines indicate which vertices are connected to the central vertex (Boisvert, 2010).

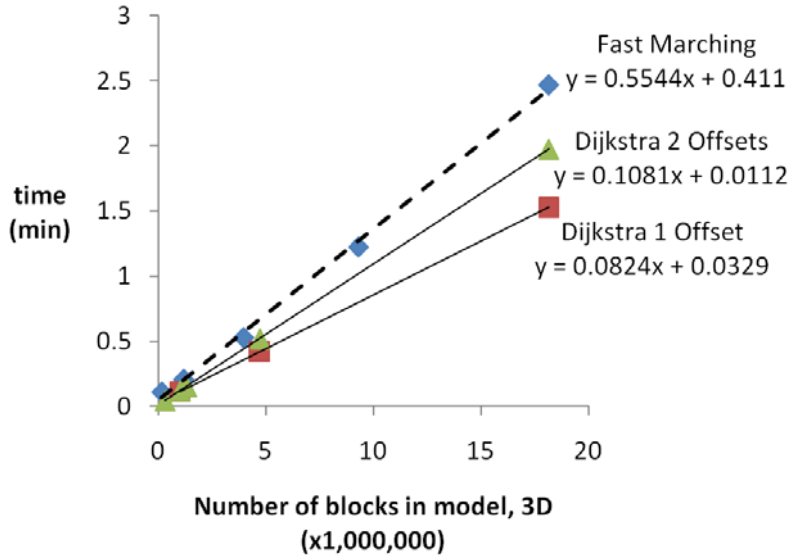

Figure 3: CPU time to calculate the distance from one landmark point to all other cells in the model.

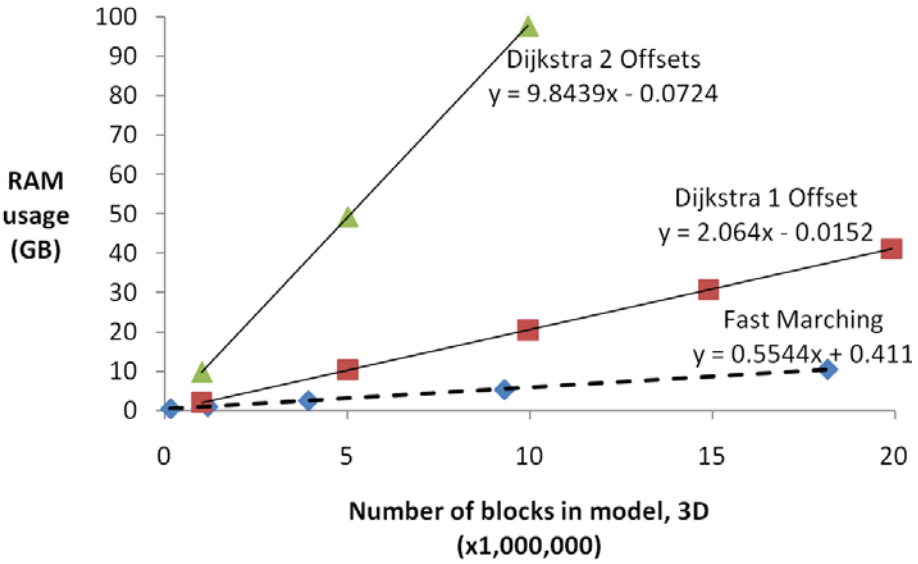

Figure 4: RAM requirements of the shortest path algorithms.

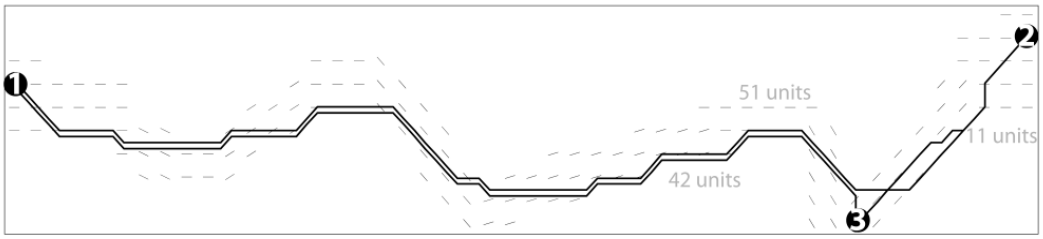

Figure 5: Path between three nodes in a channel LVA field (Boisvert 2010).

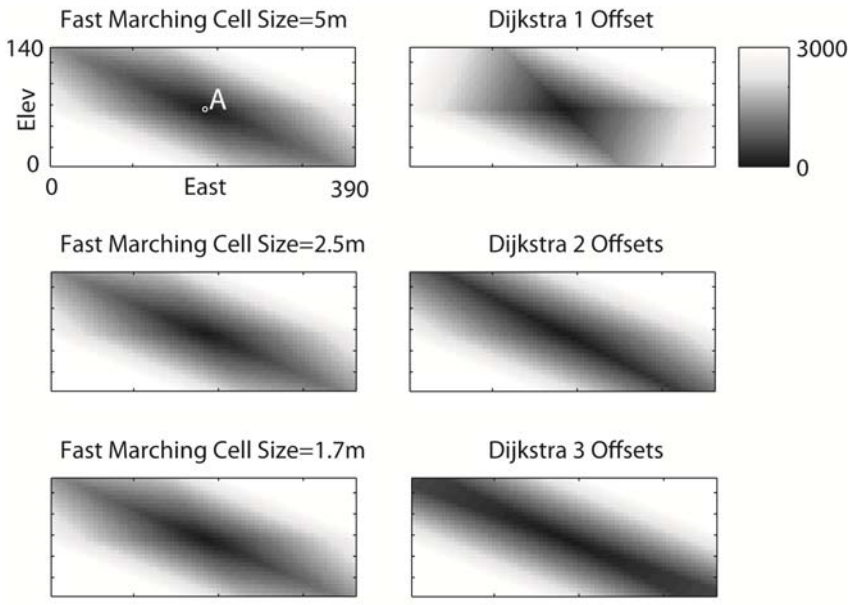

Figure 6: Dijkstra and fast marching algorithms with increasing accuracy (down). The section shows the shortest anisotropic distance from location A to all cells with a constant anisotropy of 22.5˚.

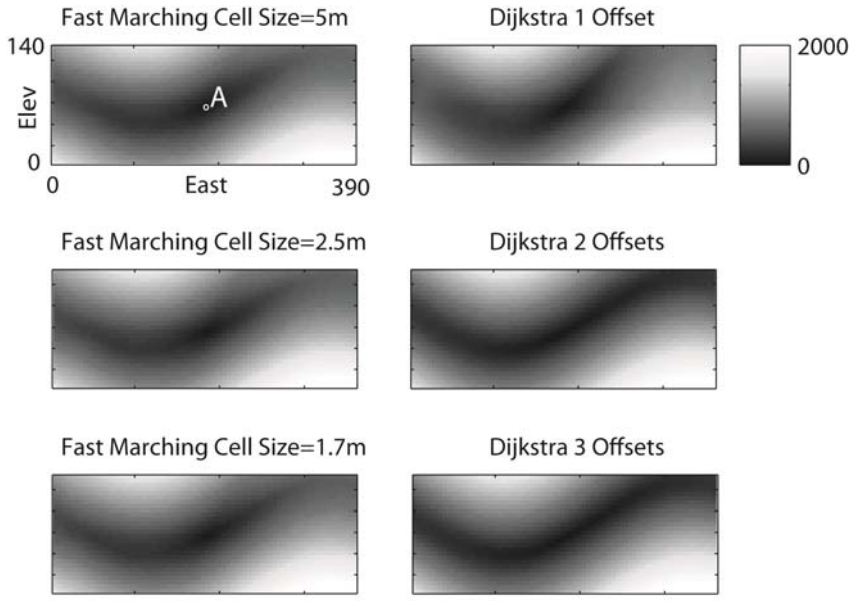

Figure 7: Dijkstra and fast marching algorithms with increasing accuracy (down). The section shows the shortest anisotropic distance from location A to all cells in the sinusoidal wave LVA field.

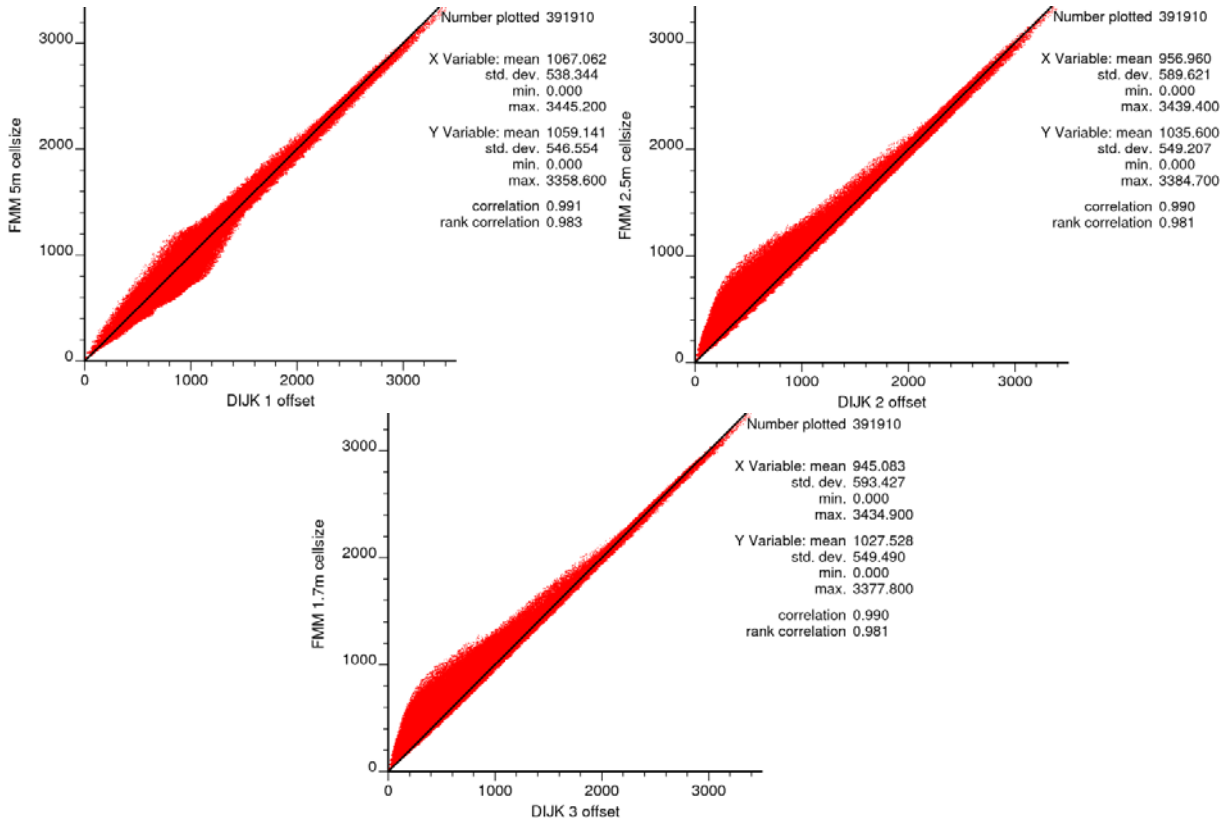

Figure 8: Comparison of the distances between 27 landmark points and all cells in the model (72x72x28 blocks). Multiple plots are generated by increasing the number of offsets (Dijkstra) and decreasing the cell size (Fast Marching).