From Everywhere To Everywhere (FETE): Adaptation of a Pedestrian Movement Network Model to a Hybrid Parallel Environment

A. Sorokine, D. White, A. Hardin

Oak Ridge National Laboratory,
One Bethel Valley Road
P.O.Box 2008, MS-6017
Oak Ridge, TN 37831-6017
Telephone: +1-865-574-4966
Email: [SorokinA,whiteda1,hardinaj]@ornl.gov

Abstract

Shortest path algorithms are hard to parallelize because they require a large number of global operations to estimate the costs of alternative routes. However, some geographic problems, such as locating archeological sites and tracking the spread of infectious diseases, demand the ability to find a large number of the shortest paths on very large graphs or grids. Here we present an approach based on the out-of-RAM Dijkstra shortest path algorithm that can be employed in hybrid massively parallel or cloud environments. In this approach we partition the graph, precompute all paths inside each partition, and then assemble the routes from precomputed paths. We demonstrate the utility of this approach by estimating travel frequency in pedestrian networks.

Keywords: single source shortest path, population movement, hybrid parallelization.

1. Introduction

High-performance parallel computers allow geoscientists to tackle hard computationally intensive problems and process large datasets. In many cases, geospatial data lends itself to parallelization through partitioning of a dataset across its geographic space. This technique of decomposing the computational process in the data domain has been used in a large number of projects including U.S. Geological Survey pRasterBlaster, PaRGO and others (Finn et al. 2012; Qin et al. 2014). However, some geoprocessing problems, like the shortest path problem, are notoriously hard to parallelize due to their reliance on a large number of global operations.

The computational requirements of finding the shortest path in a network are primarily driven by the network size. In many practical cases the network size is relatively small, although in some cases computational requirements go beyond the capabilities of an average desktop or server. Examples include evacuation planning, computing alternative routes, and modelling movement without destination that involves finding a very large number of shortest paths. In such cases the computation is limited either by the available Random-Access Memory (RAM), or, for the case of the out-of-RAM algorithms, may be restricted by an unacceptably long time to complete. These limitations can be overcome by utilizing high-performance parallel systems.

We present a strategy of porting of an application of the shortest path problem, described in White and Barber (2012), to a parallel computing environment. The purpose

of the original algorithm was to predict where prehistoric population movements were likely to be channelled, thus providing insight into ancient trade/exchange networks and where archaeological sites may be located. It also can be applied for modern problems too (fig. 1). The application works by aggregating a large set of shortest paths between many origins and destinations (From Everywhere To Everywhere, FETE). FETE finds Dijkstra's shortest path from each origin to every destination in a way similar to the Floyd-Warshall algorithm (i.e., the priority queue is filled once for each origin and a complete set of destinations). The implementation of FETE in White and Barber (2012) can utilize shared-memory parallelism by calculating all paths from a single origin in parallel threads. The major limitation of this approach is the size of the Digital Elevation Model (DEM) that can be loaded into RAM. Overcoming this limitation is the main intent of this study. By improving FETE's ability to process larger, high-resolution DEMs, we can improve the quality of the travel network prediction.

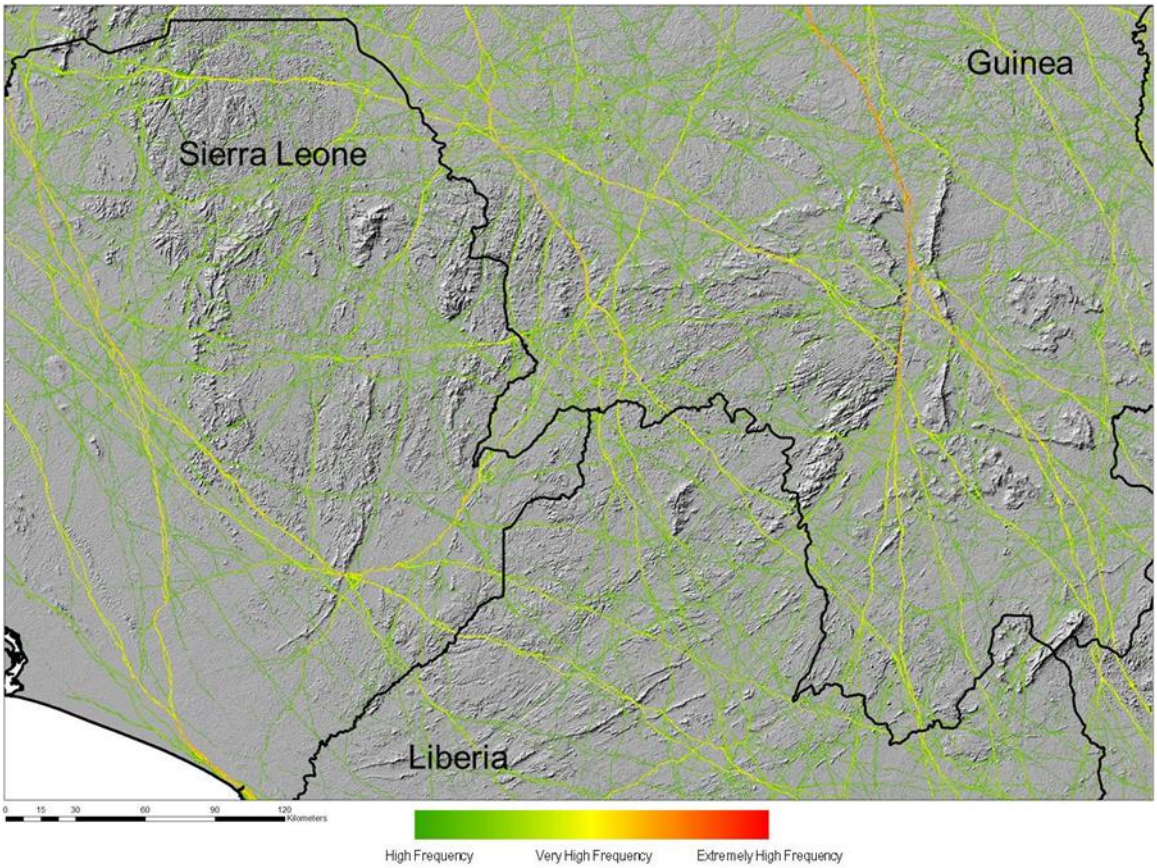

Figure 1. Computed Pedestrian Network in the Area of Ebola Outbreak. The figure shows computed pedestrian network and travel frequency overlaid on top of the shaded relief and state boundaries for the region of Ebola outbreak in West Africa.

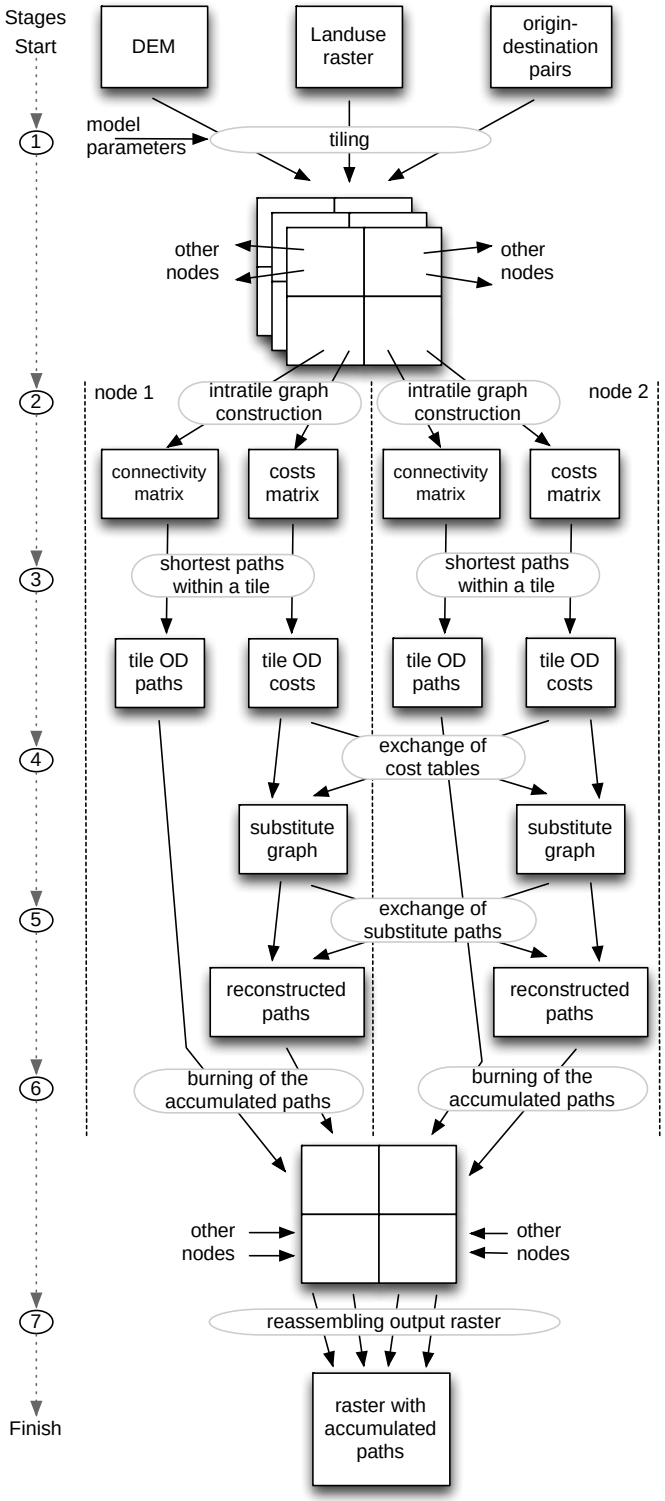

Figure 2. FETE Algorithm with Hybrid Parallelism. The algorithm partitions the DEM (stage 1), precomputes all paths inside each tile (stages 2-3), builds the substitute graph (stages 4-5), assemble the routes from precomputed paths (stage 5-6), and saves the results in the output raster (stage 7).

As stated before, we are unable to apply the technique of decomposing the problem into spatial partitions because each partition would then need to frequently update a global data structure. Several approaches for decomposing of the Dijkstra shortest path problem have been described in the literature. One approach works by splitting the priority queue into several subqueues and distributing subqueues among the computing nodes (Matloff 2006). Another approach for out-of-RAM shortest path computation works by decomposing the input graph into subgraphs, precomputing all paths between the entry and exit nodes inside each subgraph, and then computing the final route on the substitute graph built from the links between entry and exit nodes of the subgraphs (Madduri et al. 2007; Hazel et al. 2008).

For the purposes of this study we have chosen the latter approach. The main drawback of this approach is the large overhead introduced by precomputing all paths in a subgraph. However, because FETE works by aggregating a very large number of paths, the overall overhead of the path precomputation will be less than the overhead resulting from the data exchange among the nodes with subqueues.

2. Proposed Solution

Our algorithm leverages a hybrid parallelism that combines shared memory multithreaded computation using OpenMP within each node, and distributed memory computation with communication facilitated by Message Passing Interface (MPI). We process subgraphs each on a separate node in parallel. Compared to out-of-RAM version, our implementation has several optimizations to utilize parallelism on varying architectures. We support four memory management models that users can choose between depending upon their hardware configuration: 1) large data structures are kept in memory, 2) mapped to swap space, 3) offloaded to disk files, or 4) exchanged among the computing nodes using MPI one-sided communication. Furthermore, users can choose to limit the number of input/output threads to prevent the shared file system from overloading.

Our implementation of the FETE algorithm is outlined in fig. 2. First, each node loads two tiles, one from a DEM and one from a land use raster. The tile layout is provided as a model parameter, along with the physical properties of the traveller (height, weight, etc.) and the type of hiking function. As part of the initialization process, each node loads or generates a list of global origin-destination (OD) pairs. At stage 2 the connected graphs for intra-tile paths are created. Each connected graph is represented by a pair of matrices. The first matrix, connectivity, is a binary matrix that indicates existence of a link between a cell and its neighbours. The other matrix, costs, records the cost of traveling between each cell and its neighbours. During stage 3, each node uses multiple threads to calculate four sets of paths using Single-Source Shortest Path (SSSP) Dijkstra algorithm. These sets are as follows: (1) a set of paths between all the cells along the tile edge, (2) a set of paths between the edge cells of the tile and global destinations within the current tile, (3) the paths between the global origins within the current tile and destinations at the tile's edges, and (4) paths between the global origins and destinations that both fall within the current tile. The system caches these four sets of paths and their costs.

At stage 4 the tables containing path costs are exchanged among the cluster nodes, after which each computing node constructs a substitute graph. Next, the SSSP Dijkstra algorithm is run in multiple threads on the substitute graph for each global origin that

falls within the current tile. During stage 5 the computing nodes exchange the entrance and exit pairs of the paths in the substitute graph, after which they are used to reconstruct actual paths inside the tile (stage 6). Finally the count of the paths crossing each cell is increased in the accumulated paths grid and the grid is assembled into a single raster.

3. Results and Conclusions

We have successfully tested the proposed solution on a cluster and scaled the solution up to 36 computational nodes and DEMs up to about 3 million cells. One of the resulting networks is shown on fig. 1. The current version is able to process an input DEM at the rate of about one million pixels per node per hour. Our current research is aimed at scaling the system up to utilize massively parallel systems like Oak Ridge National Laboratory Titan supercomputer or cloud-based computing resources for excessively sized input DEMs.

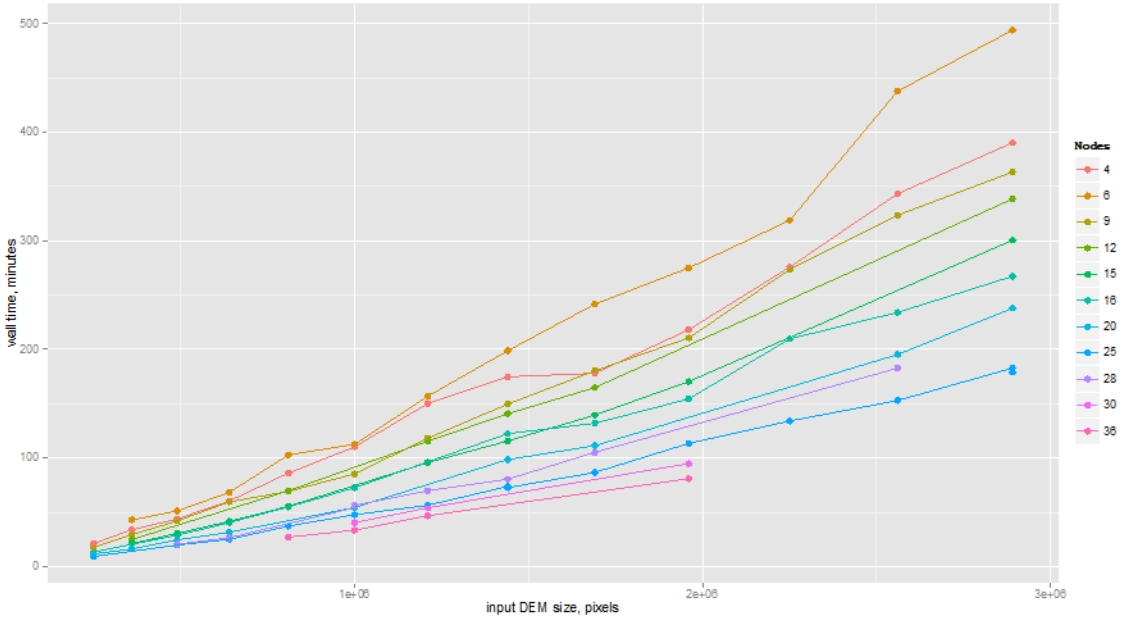

Figure 3. Performance evaluation of the algorithm

In this abstract we have presented an approach for parallelizing an application of the shortest path problem in a hybrid (multi-threaded and distributed memory) environment. We have based our approach on an existing out-of-RAM version of the shortest path algorithm that was optimized for parallel computing. Our implementation overcomes the limiting factor of available memory, and shows significantly improved performance compared to a single processor implementation.

4. Acknowledgements

This manuscript has been authored by employees of UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy. Accordingly, the United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up,

irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

5. References

Finn, MP, Y Liu, DM Mattli, Q Guan, KH Yamamoto, E Shook, and B Behzad. 2012. "pRasterBlaster: High-Performance Small-Scale Raster Map Projection Transformation Using the Extreme Science and Engineering Discovery Environment." In Abstract Presented at the *XXII International Society for Photogrammetry & Remote Sensing Congress*, Melbourne, Australia.

Hazel, T, L Toma, J Vahrenhold, and R Wickremesinghe. 2008. "Terracost: Computing Least-Cost-Path Surfaces for Massive Grid Terrains." *Journal of Experimental Algorithmics* 12: 1–9.

Madduri, K, DA Bader, JW Berry, and JR Crobak. 2007. "An Experimental Study of a Parallel Shortest Path Algorithm for Solving Large-Scale Graph Instances." In 2007 *Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments*, edited by David Applegate and Gerth Stølting Brodal, 23–35. Philadelphia, PA: Society for Industrial and Applied Mathematics. http://epubs.siam.org/doi/abs/10.1137/1.9781611972870.3.

Matloff, N 2006. "Introduction to Parallel Processing." http://pdf-release.net/external/1062852/pdf-release-dot-net-ParProc.pdf.

Qin, CZ, LJ Zhan, AX Zhu, and CH Zhou. 2014. "A Strategy for Raster-Based Geocomputation under Different Parallel Computing Platforms." *International Journal of Geographical Information Science* 2014 (34): 1–18. doi:10.1080/13658816.2014.911300.

White, DA, and SB Barber. 2012. "Geospatial Modeling of Pedestrian Transportation Networks: A Case Study from Precolumbian Oaxaca, Mexico." *Journal of Archaeological Science* 39 (8): 2684–96. doi:10.1016/j.jas.2012.04.017.