

A Fast Method for Calculating Emergency Response Times Using Travel Resistance Surfaces¹

**David C. Hatfield,² Marc R. Wiitala,³ Andrew E. Wilson,⁴
Eric J. Levy⁵**

Abstract

An algorithm is described that quickly calculates minimum travel times between locations for initial response forest fire suppression units. The algorithm was developed for integration into wildfire planning simulation models to quickly identify fire suppression units with the fastest response to a fire during a simulation. While generally quite fast, network-based path minimization algorithms are not suitable for the wildfire problem since fires are often located away from existing network features, such as roads and trails. On the other hand, traditional raster-based least-cost path algorithms can take minutes, if not hours, to calculate each resource's travel time, too slow to be effective within a simulation model. The algorithm presented in this paper enhances the raster-based technology, providing a multi-resolution approach. Our test results show the new algorithm dramatically increases the speed of calculations with only limited loss of accuracy

Introduction

One vexing problem confronting the design of realistic simulation models for initial attack wildfire planning is accurately estimating ground suppression resource travel times between locations. From the standpoint of initial attack simulation model building, there is more than just a need for accurate estimates of travel times; there is also a need within the simulation model to quickly make these calculations thereby maintaining reasonable simulation run times.

In this paper, we describe a computationally efficient shortest-path algorithm for computing travel times within an initial attack wildfire simulation model. This algorithm, COSTPATHSQ (Q for quick), was developed to meet the needs of the Wildfire Initial Response Assessment System (WIRAS), a comprehensive wildfire initial attack planning tool (Wiitala and Wilson, this proceedings).

¹ An abbreviated version of this paper was presented at the second international symposium on fire economics, planning, and policy: a global view, 19–22 April 2004, Córdoba, Spain.

² GIS Developer, Digital Visions Enterprise Unit, Forest Service, U.S. Department of Agriculture, 333 SW 1st Ave, Portland, OR, 97204.

³ Operations Research Analyst, Forestry Sciences Laboratory, Forest Service, U.S. Department of Agriculture, 620 SW Main Street, Suite 400, Portland, OR 97205.

⁴ Resource Information Specialist, Forestry Sciences Laboratory, Pacific Southwest Research Station, U.S. Department of Agriculture, 620 SW Main Street, Suite 400, Portland, OR 97205.

⁵ GIS Analyst, Digital Visions Enterprise Unit, Forest Service, U.S. Department of Agriculture, 1230 NE Third St., Suite A-262, Bend, OR 97701.

WIRAS simulates the effectiveness of a ground and air resource suppression organization in fighting forest wildfires. During a simulation, WIRAS needs to evaluate the opportunities for dispatching up to 100 suppression resources to numerous fire locations. This need for evaluation can happen many times during a single simulation leading to tens of thousands of travel time estimates. To be effective, the WIRAS simulation model requires computations to occur within a fraction of a second while providing a limited loss of accuracy on estimates of travel time. To accomplish this objective, we first explored the possibility of computing travel times over a network using the Dijkstra shortest-path algorithm (Dijkstra 1959). While numerous enhancements have been made over the years to the Dijkstra algorithm (Zahn 1997), for our particular problem, tests using the Dijkstra algorithm as implemented within ArcInfo (ESRI 2001) proved much too slow. Therefore, we decided to develop COSTPATHSQ, a new variation on the implementation of the Dijkstra algorithm that could exploit the particular nature of our travel time problem and meet the computational requirements for the WIRAS simulation model.

Methods

COSTPATHSQ uses the Dijkstra algorithm to estimate the least-cost path, where cost refers to travel time in this instance. In our model, travel is performed on a raster cost surface or grid rather than a vector network because wildfires do not necessarily fall on network features, such as roads and trails. A network could be constructed across the whole landscape, but the file size would be excessive and we believe the processing time would be prohibitively slow. A raster cost surface can be viewed as a regular triangular network with a network node at the center of every cell in the raster and network links radiating from each node to each of the eight adjacent nodes, horizontally, vertically, and diagonally. The main enhancement to the Dijkstra algorithm described here is the construction and employment of a multi-resolution cost surface, or cost pyramid. The cost pyramid allows COSTPATHSQ to initially and quickly move across a coarse resolution surface. The algorithm employs finer resolution surfaces to more accurately locate the end points (*fig. 1*). Depending on computational tolerance and accuracy needs, the user of COSTPATHSQ determines the number successive levels of resolution desired.

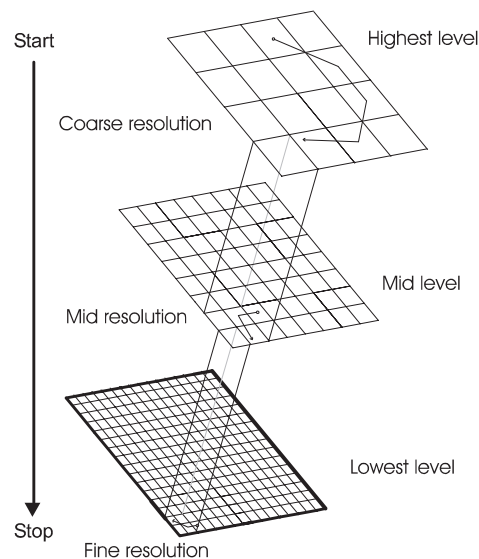


Figure 1—How least-cost paths are constructed using a cost pyramid.

Cost Pyramid

The COSTPATHSQ algorithm determines for a cost surface the least-cost path between points using the Dijkstra algorithm in a manner similar to its implementation in ArcInfo (ESRI 2001). As mentioned above, COSTPATHSQ enhances the Dijkstra algorithm by employing the use of a multi-resolution cost surface, or cost pyramid. The cost pyramid is constructed by successively aggregating blocks of four cells in an upward progression through the surfaces (*fig. 1*). The first or bottom level of the pyramid is constructed of the original cells in the cost surface. These cells are grouped into fours to form new cells on the second level of the pyramid. Within each second-level cell, the first-level cell that is deemed to be the best pathway is selected as the center or hub of the second-level cell. The cost to travel between adjacent second-level cells is calculated between the cells' hubs using the Dijkstra algorithm to travel on the first level of the pyramid. These eight travel costs are stored in each of the cells in the second level of the pyramid. Subsequently the second-level cells are grouped into fours to form cells on the third level of the pyramid (the highest level in *fig. 1*). Within each third-level cell, the second-level cell that is deemed to be the best pathway is selected as the hub, and the cost between third-level cells is calculated using the Dijkstra algorithm to travel on the second level of the pyramid. This process is repeated until only two rows and/or columns of cells remain. The result is increasingly generalized cost surfaces that retain much of the accuracy of the source data. This system of cost surfaces allows COSTPATHSQ to estimate the least-cost path between points in a fraction of a second with a limited loss of accuracy.

Pathways

The determination of which cells make better pathways is done prior to constructing the pyramid and is based upon the underlying transportation network. It is accomplished by selecting the one cell from the entire raster that bears the lowest cost and traveling the least-cost path to this origin cell from every other cell in the cost surface. If more than one cell bears the lowest cost, then one of these cells is selected at random to be the origin cell. The result of this preprocessing is a pathway grid. The value of each cell in the pathway grid is the number of times that that cell is crossed when traveling from each cell in the grid to the origin cell. Cells that receive higher values are deemed to be better pathways. Thus when four first-level cells are grouped to form a second-level cell, the first-level cell with the highest value in the pathway grid is selected as the hub of the second-level cell. This maximum value is assigned to the second-level cell and is used during the determination of hubs on the third-level of the pyramid, and so on.

Least-Cost Path

Once the cost pyramid has been constructed, it can be used to quickly estimate the least-cost path between two points. The user selects which level of the cost pyramid to use for the analysis. Traveling on higher levels is faster but less accurate. Operating on a selected level, the least-cost path between the source and target points is determined using the Dijkstra algorithm. However, the path stops one cell short of the source and target points (*fig. 1*). This is necessary because the target points may not fall on the hub of the cell, and traveling to the hub of the cell may result in backtracking and thus an overestimation of cost. Instead, a least-cost path analysis

between these end points and the hubs of the next to last cells in the path are performed on the next lower level of the pyramid, again stopping one cell short of the destination. This process is repeated down to the lowest level of the pyramid. Finally, the cost, and optionally the actual path, is output to a file.

Test Procedures

The Umatilla National Forest in Eastern Oregon and Washington in the northwestern United States was selected as the site to test the speed and accuracy of the COSTPATHSQ algorithm. At this site, a cost surface of travel times was constructed where the value of each cell in the grid was set equal to the fraction of a minute it would take for fire suppression resources to travel one meter within the cell. The cost surface was constructed based on on-road and off-road travel speeds (Wilson and Wiitala 2003). The size of each cell was 30 meters on a side. In the whole grid there were over 8,000 rows and columns of cells.

We conducted our primary test on a subset of the grid containing 2,169 rows and 2,235 columns, about 4,400 square kilometers in area. To determine how well the COSTPATHSQ algorithm would perform on smaller and larger datasets, we also performed tests on 655 by 677 and 7,177 by 7,375 grids—about 390 and 46,600 square kilometer areas, respectively.

Ten source points (representing suppression resources) and 32 target points (representing wildfires) were selected at random within each of the three Umatilla datasets. In each test, this required determining 320 travel times between the sets of points.

Travel times were also calculated using the standard Dijkstra algorithm from the Grid module of ArcInfo 8.1 to produce a baseline with which to compare the results from COSTPATHSQ. Tests were conducted on a computer with a 1.0 GHz processor, 512 MB of RAM, and a Windows 2000 operating system.

Results

We report the full results of the test for the mid-sized dataset only (*table 1*), and briefly comment on performance in the smaller and larger datasets. As our benchmark for performance comparisons, the Dijkstra algorithm, which produces optimal results, took 2,500 seconds to find the minimum travel time paths for the 320 combinations of source and target points. By contrast, running the COSTPATHSQ algorithm at level 9 of the pyramid, the coarsest level of resolution, took only 16 seconds (.05 second per path) and was 160 times faster than Dijkstra. However, at this level of resolution there was a 15 percent average overestimate of travel times relative to the Dijkstra algorithm. Running COSTPATHSQ at successively lower levels of the pyramid (finer grid resolutions) improves accuracy, but only at the expense of slower calculation times. At pyramid level 5, accuracy improved by almost five times with only a doubling of the run time (*table 1*). Below level 5, gains in accuracy came at the expense of increasingly greater run times. In this instance COSTPATHSQ did not yield results on levels 1 and 2 of the pyramid because the computer used to run the tests did not have enough memory.

Table 1--A comparison of COSTPATHSQ and Dijkstra test results.

	Dijkstra	Pyramid build	COSTPATHSQ								
			Pyramid level								
			9	8	7	6	5	4	3	2	1
Run time (sec)	2,500	2,200	16	18	19	21	32	84	490	-	-
Relative speed ¹	1.0		160	140	130	120	78	30	5.1	-	-
Accuracy ²	.00		15	11	8.6	4.8	3.1	1.5	.67	-	-

¹Magnitude faster than the basic Dijkstra algorithm.

²Percent by which travel time is overestimated.

Before leaving these results, we wish to point out there is an up front computational cost to using the COSTPATHSQ algorithm. In this test, it took 2,200 seconds to build the cost pyramid, nearly as long as running the Dijkstra algorithm (table 1). However, after building the pyramid, COSTPATHSQ achieves tremendous performance gains over the standard Dijkstra algorithm. Our tests with the smaller and larger datasets provided some interesting observations. The relative performance advantage of COSTPATHSQ increases with the size of the dataset. For example, if it is acceptable to overestimate the least cost by five percent, our tests with the smaller dataset showed COSTPATHSQ to be only about 12 times as fast as Dijkstra. In the mid-sized dataset COSTPATHSQ was 120 times as fast. Moreover, our tests with the large dataset showed COSTPATHSQ to be about 1,500 times faster than Dijkstra. Our experimentations also showed the performance of COSTPATHSQ relative to Dijkstra to increase with greater numbers of source and target points.

Conclusions

COSTPATHSQ has successfully opened the door to making more realistic and computationally fast travel time calculations thereby improving initial attack forest fire simulation models. This algorithm under most circumstances is much quicker than the standard Dijkstra algorithm in calculating suppression resource response travel times in the forest wildfire environment. Using a multi-resolution approach, our test results show COSTPATHSQ to yield fast results with a limited loss of accuracy. We also found the benefit of using COSTPATHSQ to be greater when dealing with larger datasets. Furthermore, it enables the user to select the resolution of the analysis, so they can strike an acceptable balance between speed and accuracy.

Acknowledgements

The multi-resolution approach to least-cost path analysis was inspired by Alan Ager, Pacific Northwest Research Station, USDA Forest Service, La Grande, Oregon.

References

- Dijkstra, E.W. 1959. **A Note on Two Problems on Connection with Graphs**. *Numerische Mathematik*, 1:269:271.
- Environmental Systems Research Institute. 2001. **Cell-based Modeling with GRID-Weighted Distance Functions**.
- Wilson, Andrew E., and Marc R. Wiitala. 2003. **Estimating Travel Times to Forest-Fires Using Resistance Surfaces**. Twenty-Third Annual ESRI International User Conference Proceedings, 7-11 July 2003, San Diego, California: Environmental Systems Research Institute, Inc., Redlands, California; unpaginated CD-ROM.
- Zahn, F. Benjamin. 1997. **Three Fastest Shortest Path Algorithms on Real Road Networks: Data Structures and Procedures**. *Journal of Geographic Information and Decision Analysis* 1(1):69–82.