

# Un Método Rápido para Calcular los Tiempos de Respuesta de Emergencia Utilizando Superficies de Resistencia al Viaje<sup>1</sup>

David C. Hatfield,<sup>2</sup> Marc R. Wiitala,<sup>3</sup> Andrew E. Wilson,<sup>4</sup> Eric J. Levy<sup>5</sup>

## Resumen

Se describe un algoritmo que calcula rápidamente los tiempos mínimos de viaje entre lugares para unidades de extinción de incendios forestales de respuesta inicial. El algoritmo se desarrolló para integración en modelos de simulación de planificación de incendios incontrolados para identificar rápidamente las unidades de extinción de incendios con la respuesta más rápida a un incendio durante una simulación. Aunque en general son bastante rápidos, los algoritmos de minimización de rutas basados en red, no son adecuados para el problema de incendios incontrolados ya que dichos incendios están situadas con frecuencia lejos de características existentes de la red, tales como carreteras y pistas forestales. Por otra parte, los algoritmos tradicionales de la ruta del menor coste basados en retícula pueden tardar minutos, cuando no horas, para calcular el tiempo de viaje de cada recurso, lo que resulta demasiado lento para ser efectivos dentro de un modelo de simulación. El algoritmo presentado en esta ponencia mejora la tecnología basada en retícula proporcionando un planteamiento multirresolución. Nuestros resultados de prueba demuestran que el nuevo algoritmo aumenta espectacularmente la velocidad de los cálculos con una pérdida de precisión muy limitada.

## Introducción

Un problema molesto al que tiene que hacer frente al diseño de modelos realistas de simulación para la planificación del ataque inicial de incendios incontrolados es la estimación exacta de los tiempos de viaje de los recursos de extinción en tierra de un lugar a otro. Desde el punto de vista de la creación del modelo de simulación de ataque inicial, se necesita algo más que estimaciones exactas de los tiempos de viaje, ya que también es necesario, dentro del modelo de simulación, hacer rápidamente

---

<sup>1</sup> Una versión abreviada de esta ponencia se presentó en el segundo simposio internacional sobre economía, política y planificación de incendios: una visión global, 19–22 de Abril de 2004, Córdoba, España.

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

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

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

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

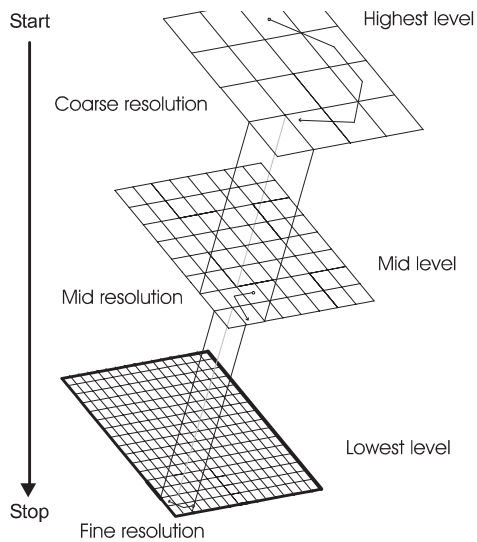
estos cálculos para conseguir así mantener unos tiempos de ejecución de la simulación razonables.

En esta ponencia, describimos un eficiente algoritmo de la ruta más corta basado en ordenador para el cálculo de los tiempos de viaje dentro de un modelo de simulación de ataque inicial a incendios incontrolados. Este algoritmo, COSTPATHSQ (Q por rápida), se desarrolló para satisfacer las necesidades del Wildfire Initial Response Assessment System (WIRAS) (sistema de evaluación de la respuesta inicial a incendios incontrolados), que es una amplia herramienta de planificación de ataque inicial a incendios incontrolados (Wiitala y Wilson, en estas actas).

WIRAS simula la efectividad de una organización de extinción terrestre y aérea en la lucha contra incendios forestales incontrolados. Durante una simulación, WIRAS tiene que evaluar las oportunidades para enviar hasta 100 recursos de extinción a numerosos lugares de incendio. Esta necesidad de evaluación puede presentarse muchas veces durante una sola simulación, lo que conduce a decenas de miles de estimaciones de tiempos de viaje. Para ser efectivo, el modelo de simulación WIRAS requiere que los cálculos se realicen en una fracción de segundo, proporcionando al mismo tiempo una pérdida de precisión limitada en las estimaciones del tiempo de viaje. Para alcanzar ese objetivo, exploramos en primer lugar la posibilidad de calcular los tiempos de viaje a través de una red utilizando el algoritmo de la ruta más corta de Dijkstra (Dijkstra, 1959). Aunque se han introducido numerosas mejoras a lo largo de los años en el algoritmo de Dijkstra (Zahn 1997), para nuestro problema concreto, las pruebas realizadas utilizando el algoritmo de Dijkstra de acuerdo con su implantación en ArcInfo (ESRI 2001) demostraron ser demasiado lentas. Por tanto, decidimos desarrollar COSTPATHSQ, una nueva variante de la implantación del algoritmo de Dijkstra que podría aprovechar la naturaleza especial de nuestro problema de tiempos de viaje y satisfacer los requisitos de cálculo para el modelo de simulación WIRAS.

## Métodos

En COSTPATHSQ se utiliza el algoritmo de Dijkstra para estimar la ruta del menor coste, entendiendo en este caso por coste el tiempo de viaje. En nuestro modelo, el viaje se realiza en una superficie de costes reticulada o retícula en lugar de realizarse en una red vectorial debido a que los incendios incontrolados no se producen necesariamente en características de la red, como carreteras y pistas forestales. Se podría construir una red a través de todo el paisaje, pero el tamaño del archivo sería excesivo y creemos que el tiempo de procesamiento sería prohibitivamente lento. Una superficie de costes reticulada se puede considerar como una red de triángulos regulares con un nodo de red en el centro de cada celda de la retícula y enlaces de red radiando desde cada nodo a cada uno de los ocho nodos adyacentes en horizontal, vertical y diagonal. La principal mejora respecto al algoritmo de Dijkstra descrita en esta ponencia, es la construcción y empleo de una superficie de costes multirresolución o pirámide de costes. La pirámide de costes permite que COSTPATHSQ se mueva inicialmente de forma rápida a través de una superficie de resolución basta. El algoritmo utiliza superficies de resolución más finas para localizar con más precisión los puntos finales (figura 1). Dependiendo de la tolerancia de los cálculos y de las necesidades de precisión, el usuario de COSTPATHSQ determina el número de niveles sucesivos de resolución deseados.



**Figura 1**— Manera de construir rutas del menor coste utilizando una pirámide de costes.

### **Pirámide de costes**

El algoritmo COSTPATHSQ determina para una superficie de costes la ruta del menor coste entre puntos utilizando el algoritmo de Dijkstra de forma similar a como se utiliza en su implantación en ArcInfo (ESRI 2001). Como se ha mencionado anteriormente, COSTPATHSQ mejora el algoritmo de Dijkstra mediante el uso de una superficie de costes multirresolución o pirámide de costes. La pirámide de costes se construye agregando sucesivamente bloques de cuatro celdas en progresión ascendente a través de las superficies (figura 1). El primer nivel o nivel inferior de la pirámide se construye a partir de celdas originales de la superficie de costes. Estas celdas se agrupan de cuatro en cuatro para formar nuevas celdas en el segundo nivel de la pirámide. Dentro de cada celda del segundo nivel, la celda del primer nivel que se considera que es la mejor ruta se selecciona como el centro o núcleo de la celda del segundo nivel. El coste de viajar entre celdas adyacentes del segundo nivel se calcula entre los núcleos de las celdas utilizando el algoritmo de Dijkstra para viajar en el primer nivel de la pirámide. Estos ocho costes de viaje se almacenan en cada una de las celdas del segundo nivel de la pirámide. A continuación, las celdas del segundo nivel se agrupan de cuatro en cuatro para formar celdas del tercer nivel de la pirámide (el nivel más alto en la figura 1). Dentro de cada celda del tercer nivel, la celda del segundo nivel que se considera que es la mejor ruta se selecciona como el núcleo y el coste entre celdas del tercer nivel se calcula utilizando el algoritmo de Dijkstra para viajar por el segundo nivel de la pirámide. Este proceso se repite hasta que sólo quedan dos filas y/o columnas de celdas. El resultado son superficies de costes cada vez más generalizadas que conservan una gran parte de la precisión de los datos de origen. Este sistema de superficies de costes permite que COSTPATHSQ estime la ruta del menor coste entre puntos en una fracción de segundo y con una pérdida de precisión limitada.

### **Rutas**

La determinación de las celdas que constituyen la mejor ruta se hace antes de construir la pirámide y está basada en la red de transporte subyacente. Esto se logra

seleccionando la celda de toda la retícula a la que corresponde el menor coste y recorriendo la ruta del menor coste hasta esta celda de origen desde cada una de las demás celdas de la superficie de costes. Si el menor coste corresponde a más de una celda, se selecciona aleatoriamente una de estas celdas para que sea la celda de origen. El resultado de este procesamiento previo es una retícula de rutas. El valor de cada celda de la retícula de rutas es el número de veces que se cruza la celda cuando se viaja desde cada una de las celdas de la retícula hasta la celda de origen. Las celdas que reciben los valores más altos se considera que son las rutas mejores. De esa manera, cuando se agrupan cuatro celdas del primer nivel para formar una celda del segundo nivel, la celda del primer nivel con el valor más alto en la retícula de rutas se selecciona como el núcleo de la celda del segundo nivel. Este valor máximo se asigna a la celda del segundo nivel y se utiliza durante la determinación de los núcleos en el tercer nivel de la pirámide, etc.

## Ruta del menor coste

Una vez construida la pirámide de costes, se puede utilizar para estimar rápidamente la ruta del menor coste entre dos puntos. El usuario selecciona el nivel de la pirámide de costes que desea utilizar para el análisis. El viaje a niveles más altos es más rápido pero menos exacto. Trabajando en un nivel seleccionado, la ruta del menor coste entre los puntos de origen y destino se determina utilizando el algoritmo de Dijkstra. Sin embargo, la ruta se interrumpe una celda antes de los puntos de origen y destino (figura 1). Esto es necesario porque los puntos destino no pueden caer en el núcleo de la celda y el viaje hasta el núcleo de la celda puede dar por resultado un retroceso y por consiguiente una sobreestimación del coste. En lugar de eso, se realiza un análisis de la ruta del menor coste entre esos puntos extremos y los núcleos de las celdas siguientes hasta la última de la ruta en el siguiente nivel más bajo de la pirámide, interrumpieron o de nuevo el viaje una celda antes del destino. Este proceso se repite en sentido descendente hasta llegar al nivel más bajo de la pirámide. Finalmente, el coste, y opcionalmente la ruta real, se envía a un archivo.

## Procedimientos de prueba

El bosque nacional Umatilla situado en la zona oriental de Oregón y Washington en el noroeste de Estados Unidos se seleccionó como el lugar para probar la velocidad y precisión del algoritmo COSTPATHSQ. En ese lugar, se construyó una superficie de costes de tiempos de viaje en la que el valor de cada celda de la retícula se hizo igual a la fracción de minuto que tardarían los recursos de extinción de incendios en viajar un metro dentro de la celda. La superficie de costes se construyó sobre la base de velocidades de viaje en carretera y fuera de la carretera (Wilson y Wiitala 2003). El lado de cada celda tenía 30 metros. En la retícula completa había más de 8.000 filas y columnas de celdas.

Realizamos nuestra prueba primaria en un subconjunto de la retícula que contenía 2.169 filas y 2.235 columnas y cubría aproximadamente una superficie de 4.400 kilómetros cuadrados. Para determinar hasta qué punto sería aceptable el comportamiento del algoritmo COSTPATHSQ en conjuntos de datos menores y mayores, también realizamos pruebas en retículas de 655 por 677 y 7.177 por 7.375—superficies aproximadas de 390 y 46.600 kilómetros cuadrados, respectivamente.

Dentro de cada uno de los tres conjuntos de datos de Umatilla, se seleccionaron aleatoriamente diez puntos de origen (que representaban recursos de extinción) y 32 puntos destino (que representaban incendios incontrolados). En cada prueba, esto exigió determinar 320 tiempos de viaje entre los conjuntos de puntos.

Los tiempos de viaje se calcularon también utilizando el algoritmo estándar de Dijkstra a partir del módulo de retícula de ArcInfo 8.1 para obtener una línea de base con la cual se pudieran comparar los resultados de COSTPATHSQ. Se realizaron pruebas en un ordenador con un procesador a 1,0 GHz, 512 MB de RAM y un sistema operativo Windows 2000.

## Resultados

Informamos sobre los resultados concretos de la prueba solamente para el conjunto de datos de tamaño medio (tabla 1) y comentamos brevemente el comportamiento en los conjuntos de datos menores y mayores. Como referencia para comparaciones de comportamiento, el algoritmo de Dijkstra, que proporciona resultados óptimos, tardó 2.500 para calcular las rutas de tiempos de viaje mínimo para las 320 combinaciones de puntos de origen y destino. Por contraste, ejecutando el algoritmo COSTPATHSQ al nivel nueve de la pirámide, el nivel de resolución más basta, sólo se tardaron 16 segundos (0,05 segundos por ruta) y fue 160 veces más rápido que el de Dijkstra. Sin embargo, a este nivel de resolución, se produjo una sobreestimación media del 15% de los tiempos de viaje respecto a los obtenidos con el algoritmo de Dijkstra. Ejecutando COSTPATHSQ a niveles cada vez más bajos de la pirámide (resoluciones de la retícula más finas) se mejora la precisión, pero sólo a expensas de tiempos de cálculo más largos. Al nivel cinco de la pirámide, la precisión se multiplicó casi por 5, duplicando solamente el tiempo de ejecución (tabla 1). Por debajo del nivel cinco, se obtuvieron mejoras de la precisión a expensas de tiempos de ejecución cada vez más largos. En ese caso, COSTPATHSQ no proporcionó resultados a los niveles 1 y 2 de la pirámide porque el ordenador utilizado para realizar las pruebas no tenía memoria suficiente.

**Tabla 1--** Una comparación de los resultados de las pruebas de COSTPATHSQ y Dijkstra.

	Dijkstra	Construcción de la pirámide	COSTPATHSQ								
			Nivel de la pirámide								
			9	8	7	6	5	4	3	2	1
Tiempo de ejecución (s)	2,500	2,200	16	18	19	21	32	84	490	-	-
Velocidad relativa <sup>1</sup>	1.0		160	140	130	120	78	30	5.1	-	-
Precisión <sup>2</sup>	.00		15	11	8.6	4.8	3.1	1.5	.67	-	-

<sup>1</sup>Veces más rápido que el algoritmo básico de Dijkstra.

<sup>2</sup>Porcentaje de sobreestimación del tiempo de viaje.

Antes de abandonar estos resultados deseamos señalar que hay un coste informático inicial en el uso del algoritmo COSTPATHSQ. En esta prueba, se tardaron 2.200 segundos en construir la pirámide de costes, casi tanto como el tiempo de ejecución del algoritmo de Dijkstra (tabla 1). Sin embargo, después de construir la pirámide, COSTPATHSQ consigue enormes mejoras de rendimiento respecto al algoritmo estándar de Dijkstra. Nuestras pruebas con los conjuntos de datos menores y mayores permite hacer algunas observaciones interesantes. La ventaja relativa en prestaciones de COSTPATHSQ aumenta con el tamaño del conjunto de datos. Por ejemplo, si se considera aceptable sobreestimar el menor coste en un cinco por ciento, nuestras pruebas con el conjunto de datos más pequeño mostraron que COSTPATHSQ sólo es aproximadamente 12 veces más rápido que Dijkstra. Con el conjunto de datos de tamaño medio, COSTPATHSQ resultó ser 120 veces más rápido. Además, nuestras pruebas con el conjunto de datos grande demostraron que COSTPATHSQ es aproximadamente 1.500 veces más rápido que Dijkstra. Nuestros experimentos demostraron también que las prestaciones de COSTPATHSQ respecto a Dijkstra aumentan cuando es mayor el número de puntos de origen y destino.

## Conclusiones

COSTPATHSQ ha abierto con éxito la puerta a la posibilidad de hacer cálculos más realistas y más rápidos de los tiempos de viaje, mejorando así los modelos de simulación de ataque inicial de incendios forestales. Este algoritmo, en la mayoría de los casos, es mucho más rápido que el algoritmo estándar de Dijkstra en el cálculo de los tiempos de viaje de respuesta de recursos de extinción en el entorno de incendios forestales incontrolados. Utilizando un planteamiento multirresolución, los resultados de nuestras pruebas demuestran que COSTPATHSQ proporciona resultados con mayor rapidez y con una pérdida de precisión limitada. También hemos encontrado que la ventaja de utilizar COSTPATHSQ se hace mayor cuando se manejan conjuntos de datos más grandes. Además, permite al usuario seleccionar la resolución del análisis, lo que hace posible encontrar un equilibrio aceptable entre velocidad y precisión.

## Reconocimientos

El planteamiento multirresolución del análisis de la ruta del menor coste ha sido inspirado por Alan Ager, Pacific Northwest Research Station, USDA Forest Service, La Grande, Oregon.

## Bibliografía

- 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.