Evaluación de un sistema de búsqueda de rutas de evacuación eficientes de un establecimiento usando el algoritmo D estrella (D*)

Walter Steven Pariona-Sánchez
20140989@aloe.ulima.edu.pe / Universidad de Lima, Perú

Recepción: 16-6-2019 / Aceptación: 9-7-2019

Resumen. Los desastres naturales como movimientos telúricos han generado interés en varios autores alrededor del mundo sobre el desarrollo de diferentes soluciones relacionadas a sistemas de evacuación. En esta investigación se expone la importancia de implementar un sistema de evacuación inteligente que reconozca la ruta más corta ante un movimiento telúrico real. De esta manera, la investigación llevó a cabo el proceso de construcción de un simulador para encontrar el camino más corto utilizando el algoritmo D estrella. En esta evaluación se midió el tiempo experimental que le tomó al algoritmo encontrar una ruta de evacuación eficiente bajo diferentes entornos al variar el tamaño del establecimiento, la cantidad de obstáculos iniciales y la cantidad de obstáculos colocados en tiempo real. Los resultados del simulador fueron favorables, puesto que logró identificar una ruta eficiente en 22 milisegundos y un recálculo de ruta en 3 milisegundos, para los casos en que se presenten obstáculos que interfieran con el recorrido inicial.

Palabras clave: sistemas de evacuación, algoritmos de búsqueda, algoritmo D estrella, caminos más cortos, obstáculos dinámicos

Evaluation of an Efficient Evacuation Route Search System Within an Establishment Using the D Star Algorithm

Abstract. Natural disasters such as earthquakes have aroused great interest among several authors around the world, giving rise to different solutions related to evacuation systems. This research presents the importance of implementing an intelligent evacuation system that recognizes the shortest evacuation route in a real earthquake. Based on this, the research aimed to build a simulator to find the shortest evacuation route using the D star algorithm. In this evaluation, the experimental time taken by the algorithm to find an efficient evacuation route was measured within establishments of different sizes with varying quantities of initial obstacles and varying quantities of obstacles placed in real time. The results of the simulator were favorable since it found an efficient route in 22 milliseconds and a recalculated route in 3 milliseconds for the cases in which obstacles interfered with the initial route.

Keywords: evacuation systems, search algorithms, D star algorithm, shortest evacuation routes, dynamic obstacles

1. INTRODUCCIÓN

En el Perú, los diferentes tipos de acontecimientos relacionados a un siniestro o a una situación de emergencia pueden ocurrir en cualquier momento debido a que es parte del conglomerado de países que se encuentran en el área del cinturón del fuego la cual causa el 90 % de todos los movimientos telúricos alrededor del mundo.

Adicionalmente, el 60 % del sector construcción en el Perú no ha presentado el plan de evacuación al Instituto Nacional de Defensa Civil (INDECI) (Cornejo, 6 de junio de 2015). Esto evidencia la falta de concientización de las organizaciones y su indiferencia por difundir los conocimientos relacionados a evacuaciones seguras.

Por otro lado, aunque existen establecimientos que cuentan con planes de evacuación regularizados por la institución pertinente, esos planes de escape no necesariamente son efectivos, ya que están compuestos solo de señaléticas las cuales no aseguran un proceso de evacuación eficiente en el momento de un simulacro o en un acontecimiento real.

En consecuencia, al no dejar un escenario de evacuación claro y conocido por todas las personas, estas, en un hipotético caso, irán al primer punto de salida que encuentren (Ying, Zi-min, y Jian, 2017). Además, debe destacarse que es muy probable que un individuo siga a la multitud. Esta probabilidad aumenta cuando una persona no logra identificar la salida y percibe a un flujo de personas dirigiéndose hacia una señal. Esto se debe a que los individuos tienden a creer y seguir otras decisiones de evacuadores (Haghani y Sarvi, 2016).

Debido a lo expuesto, surge el principal problema: los planes o guías de evacuación actuales muestran guías de recorrido y caminos confiables hacia los puntos de salida seguros del establecimiento; no obstante, estos planes no son óptimos al identificar la menor distancia de forma dinámica y no presentan rutas alternas en caso de que aparezcan obstáculos en el recorrido.

La presente investigación propone generar un sistema de evacuación mediante la creación de un simulador que brinde la ruta de escape más eficiente y segura a un área fuera de peligro usando el algoritmo de búsqueda de caminos más cortos, *D estrell*a (*D**).

2. ESTADO DEL ARTE

La implementación de sistemas que ayuden a buscar rutas de evacuación eficientes para los establecimientos ha sido de interés para otros autores alrededor del mundo, es decir, existe una preocupación acerca de qué se está haciendo por evitar daños y pérdidas lamentables, principalmente humanas, frente a acontecimientos como son los movimientos telúricos. Esta inquietud ha generado soluciones para abordar el problema relacionado a los planes de evacuación habituales.

2.1 Aplicaciones de sistemas de evacuación en diferentes entornos

2.1.1 Koo, Kim, Kim y Christensen (2013)

Desde el ataque terrorista del 2001 en Estados Unidos, la evacuación en eventos como ataques terroristas, incendios y sismos es centro de atención de profesionales. Sin embargo, la mayoría de los estudios consideran solo poblaciones homogéneas dejando de lado a personas con discapacidades motrices y sensoriales que necesitan de ayuda para evacuar. Además, estos incluso podrían bloquear la evacuación de otras personas debido a su movilidad reducida y el espacio requerido (Koo, Kim, Kim y Christensen, 2013).

Por ello, se propuso el modelo de simulación BUMPEE. Este tiene la capacidad de manejar distintos tipos de agentes de evacuación virtuales como agentes sin discapacidad, con sillas de ruedas, impedimento visual, entre otros. Como objeto de estudio se usó un edificio de 24 pisos diseñado con ascensores adecuados para evacuaciones (Koo *et al.*, 2013).

Debido a que en evacuaciones reales es posible que un ocupante con silla de ruedas cruce un área con la ayuda de otros individuos, el modelo de simulación implementa una rutina de "sistema de amigos" en el que agentes discapacitados reciben ayuda de otros ocupantes. Además, existe la posibilidad de que los agentes en sillas de ruedas usen los ascensores como medio de evacuación (Koo *et al.*, 2013).

En base a esta propuesta se hicieron experimentos con estrategias controladas basadas en agentes y con uso de ascensores. El primer experimento consideró 60 segundos y agentes discapacitados. El resultado mostró que todos los agentes que no usaban sillas de ruedas evacuaron más rápido respecto a una evacuación simultánea. Para el segundo experimento, los agentes discapacitados usaron ascensores. Así, los agentes terminaron la evacuación más rápida comparado a una evacuación con escaleras (Koo *et al*., 2013).

2.1.2 Hridi, Das, Anjum y Das, 2016

Bangladesh se ha visto afectada por desastres naturales como huaicos, lluvias y ciclones causando pérdidas de vidas. Debido a esto, este artículo propone una aplicación que guíe al usuario a un lugar seguro en el menor tiempo posible usando su ubicación en tiempo real y mapeando su comportamiento. Esta propuesta va dirigida para usuarios en y fuera de línea. Los usuarios en línea reciben instrucciones basadas en datos en tiempo real de otros usuarios. Según Hridi, Das, Anjum y Das (2016) el comportamiento de la aplicación para estos varía de acuerdo con la ubicación del usuario (a salvo, fuera o dentro de zona de peligro).

Es en el caso de usuarios que se encuentran fuera de línea en donde se destaca la parte innovadora de la metodología. De acuerdo con Hridi *et al.* (2016), para estos usuarios la aplicación

almacena su patrón de movimiento de los catorce últimos días y almacena información sobre la calidad de las autopistas alrededor del área en la cual se ha movido la persona.

Es así como, basada en datos de los últimos siete días, la aplicación crea un modelo de comportamiento del usuario. Usando ese modelo se predicen las rutas más propensas a ser usadas en un siniestro y se almacenan en un grafo. Así, la ruta de evacuación más segura tendrá un menor valor en el grafo. Por último, la aplicación ejecuta múltiples veces un algoritmo de búsqueda y encuentra todas las rutas de evacuación (Hridi *et al*., 2016).

Por otra parte, para usuarios sin conexión, el servidor analiza movimientos y hace una suposición de las rutas más propensas a ser usadas durante una evacuación. Así, sugerirá a usuarios en línea rutas alternativas que probablemente no sean utilizadas por usuarios sin conexión y, por lo tanto, permitirá una evacuación más rápida (Hridi *et al.,* 2016).

2.2 Investigaciones que emplean algoritmos de caminos más cortos más conocidos

2.2.1 Pelechano y Badler (2006)

Pelechano y Badler (2006) mencionan que la evacuación en multitud de edificios es usualmente obstaculizada por personas que no conocen su conectividad interna debido a que los ocupantes suelen usar las salidas familiares ignorando las salidas de emergencia. Además, el aumento del estrés genera una reducción general de la conciencia y un incremento de desorientación.

Para esto se desarrolló MACES (por sus siglas en inglés, *multi-agent communication for evacuation simulation*, es decir, comunicación multiagente para la simulación de evacuación), el cual es un sistema distribuido por *agentes* según su propio comportamiento. Cada agente puede tener una personalidad: líder entrenado, líder no entrenado y seguidor no entrenado. *Uno entrenado* va al camino más corto sin problemas, *un líder* explora el edificio usando un algoritmo de búsqueda por profundidad (DFS), *un seguidor* no sabe qué hacer y solo sigue las decisiones de otras personas (Pelechano y Badler, 2006).

MACES recibe como datos de entrada las características del ambiente como las dimensiones, el número de salidas y de agentes, el plano del edificio, el porcentaje de agentes entrenados. Luego, por cada celda del plano, el algoritmo automáticamente genera el camino más corto a cada salida (Pelechano y Badler, 2006).

En una prueba de comparación de métodos de búsqueda, el resultado del algoritmo DFS fue quince veces más rápido al buscar habitaciones en comparación con un algoritmo de búsqueda aleatoria. En una segunda prueba de comparación entre comunicación y no comunicación, el resultado de la simulación con comunicación entre agentes terminó en la mitad del tiempo que le tomó al caso sin comunicación (Pelechano y Badler, 2006).

2.3 Investigaciones que toman en cuenta el comportamiento de los evacuados

2.3.1 Iizuka y Iizuka (2015)

Proponen un sistema que funciona en los dispositivos móviles ejecutando cálculos distribuidos con el *framework* DCOP (*distributed constraint optimization problem*) el cual no necesita un servidor central. Este sistema intercambia información acera de la seguridad, además, planifica, negocia y manifiesta la ruta (Iizuka y Iizuka, 2015).

Para lograr la negociación del tiempo y usar el *framework*, se necesita realizar una formalización. Una ruta de evacuación es considerada un recurso donde cada agente tiene una variable para almacenar la ubicación a la cual debe moverse. El agente también decide la posición en el tiempo $t + 1$ usando el *framework* en el tiempo t (Iizuka y Iizuka, 2015).

Con el propósito de obtener resultados se usó un simulador de agentes múltiples. El primer experimento simulaba una evacuación en un campus de una universidad, al ejecutar esa evacuación guiada por el sistema, el tiempo de la evacuación disminuyó en un 10 % respecto a una evacuación sin sistema guiado (Iizuka y Iizuka, 2015).

3. ANTECEDENTES

Para brindar la solución planteada ante el problema expuesto es importante tener en cuenta el estado del terreno, pues debe verificarse en tiempo real si el entorno presenta obstáculos como paredes derrumbadas, personas accidentadas, etc. Estos obstáculos se reflejan en un plano virtual como el costo óptimo de un nodo y se tiene que monitorear para saber la variación de su costo en cada tiempo t y así saber si sigue siendo óptimo.

Es por esto que se presentará la lógica del algoritmo *D estrella* (D^*). Este algoritmo es un reensamblaje del algoritmo *A estrella* (A^*) para convertirse en un algoritmo dinámico ya que los parámetros del costo del arco pueden cambiar durante el proceso de ejecución.

Al inicio, el agente comienza en un estado y se mueve para alcanzar su ubicación final denotado por G. Una diferencia a recalcar del algoritmo D^* con el A^* es que cada estado, excepto G, tiene un puntero posterior (*backpointer*) a un siguiente estado Y; el costo de atravesar un arco desde Y a un estado X es un número positivo dado por la función de costo del arco $c(X, Y)$. Si Y no tiene un arco hacia X, entonces $c(X, Y)$ es indefinido (Stentz, 1994).

Cabe mencionar que D^* también mantiene una lista abierta de estados como el A^*. En esta se almacenan los cambios en la función de costo del arco y los costos de la ruta. Una característica adicional es que cada estado tiene un *tag* asociado $t(X)$, en el que $t(X) = NUEVO$ si X nunca ha estado en la lista abierta, $t(X) = ABIERTO$ si X se encuentra en la lista abierta, y $t(X) = CERRADO$ si X ya no se encuentra en la lista abierta (Stentz, 1994).

Otra característica importante del *D** es que consta de dos funciones principales: PROCESS-STATE y MODIFY-COST. La primera es usada para calcular los costos de caminos óptimos hacia el nodo final; la segunda se emplea para cambiar la función de costo de arco e ingresar los nuevos estados alterados (ocurrencia de obstáculos) a la lista abierta (Stentz, 1994).

Al iniciar, cada estado *X* tiene un *t(X) = NUEVO*, el costo de la ruta es cero y el nodo destino se inserta a la lista abierta. Luego, la función PROCESS-STATE es llamada hasta remover el estado actual de la lista o el valor –1 sea retornado lo cual significa que se encontró la ruta o no existe, respectivamente. Después de esto, el agente sigue la ruta trazada por los *back-pointers* hasta alcanzar el nodo objetivo o descubrir la presencia de un obstáculo. Seguidamente, la función MODIFY-COST corrige la función de costo del arco y actualiza la lista abierta. De esta manera, si *Y* es un estado en el que se descubre un error, nuevamente se realiza una llamada a la función PROCESS-STATE. Así, una nueva secuencia *Y* se construye y el agente sigue los punteros posteriores hasta encontrar el destino final (Stentz, 1994).

Por último, al comparar el algoritmo *D** con otros algoritmos de búsqueda, se observó que los resultados son óptimos, ya que este replantea un nuevo camino desde la última posición mientras que los otros replantean el camino como una nueva trayectoria global.

4. METODOLOGÍA

Teniendo en cuenta el objetivo y la literatura revisada, se construyó una simulación para encontrar las rutas de escape más eficientes y seguras en un área fuera de peligro implementando el algoritmo de búsqueda del camino más corto *D**.

A continuación, en la figura 1 se presenta un esquema equivalente a las etapas que fueron necesarias para la construcción de la solución propuesta.

Figura 1. Diagrama de etapas del proyecto de investigación

Elaboración propia

4.1 Proceso de generar un establecimiento en un entorno virtual

Esta es la primera etapa para comenzar a construir la solución propuesta. Se inició con la configuración inicial del entorno para lo cual se utilizó la herramienta multiplataforma Unity3D. Cabe recalcar que el uso de este *software* estuvo en todo el proceso de construir el sistema de evacuación inteligente dado que permite combinar la programación con un entorno visual.

Por otro lado, para generar el plano virtual, se guardaron las dimensiones del plano y la ubicación de cada objeto en una variable de tipo *Vector2*. Además, el sistema dividió el plano en nodos de un metro de diámetro (también llamado, plano en forma de grillas) para representar en cada nodo la ubicación del agente en el espacio.

4.2 Proceso de implementar el algoritmo de búsqueda *D**

Esta etapa es la más importante del proceso de construcción del simulador porque involucra la implementación del algoritmo, y su relación y conexión con el plano creado gráficamente en Unity. Para esto se crearon 3 *scripts* con nombre Node, GameGrid y SearchAlgorithm.

El primer *script* sirvió para representar un nodo en memoria e iniciar su posición en el espacio 3D, su posición en el plano, su identificador único, el valor de su función de costo, el valor del *tag*, del puntero posterior y un valor booleano para representar si es un obstáculo.

El segundo *script* manejó la lógica para representar el plano virtual convirtiéndolo en una grilla y encontró los nodos que no son transitables en base a obstáculos iniciales establecidos en la parte gráfica. Aquí se ejecutó el proceso de creación de nodos por cada grilla. Asimismo, una función se encargó de actualizar el plano con el objetivo de almacenar nuevos obstáculos insertados en tiempo de ejecución.

El tercer *script* fue fundamental para la obtención de la ruta óptima ya que manejó la lógica del algoritmo *D**. Como primer paso se crearon en el motor gráfico dos *GameObject* que representaron la posición del agente y la zona segura más cercana. Luego, se crearon e inicializaron dos arreglos llamados ABIERTO y CERRADO. El primero almacenó los nodos a ser evaluados; el segundo los que ya habían sido evaluados en la ejecución.

Seguido de esto, se definió una función que encontrara la ruta más corta desde la posición del agente hasta la ubicación de la zona de evacuación segura. En esta función el proceso de encontrar la ruta pasa por varias etapas en las que se ejecutan dos funciones fundamentales: *process-state()* y *modify-cost()*. A su vez, también hay funciones auxiliares encargadas de insertar, borrar, obtener el nodo con menor costo y actualizar la búsqueda debido a obstáculos. Cabe mencionar que gracias a las funciones fundamentales, el sistema fue capaz de gestionar ambientes o planos dinámicos ya que es posible insertar obstáculos en tiempo real.

4.3 Proceso de generar obstáculos dinámicos en tiempo real

Con esta etapa se implementó una fase exclusiva para la construcción de la lógica que involucra la creación dinámica y aleatoria de obstáculos sobre el plano virtual.

Para esto, además de la implementación del algoritmo de búsqueda *D**, se programó la función *createRandomObstacles()*. Esta función crea *GameObjects*, define sus posiciones y sus tamaños con base en valores generados aleatoriamente y los establece sobre el plano.

Por último, para que la función *createRandomObstacles()* se pudiera ejecutar continuamente durante la simulación, se hizo uso de la función nativa *InvokeRepeating()* la cual recibió tres parámetros. Como primer parámetro recibió la función *createRandomObstacles()* y la ejecutó cada tres segundos después de cinco segundos de iniciada la simulación.

4.4 Simular el recorrido de una persona virtual

En esta etapa el simulador comenzó su proceso de ejecución. Esto debido a que el agente virtual tuvo asignado, como comportamiento o patrón de desplazamiento, el resultado del algoritmo construido y así pudo encontrar rápidamente la ruta más adecuada.

Para entender mejor la ejecución, esta se separó en dos partes. Primero se tuvo que establecer en el plano la posición inicial del agente y la posición del punto de evacuación seguro, para luego ejecutar el simulador y así obtener el recorrido más corto desde la posición inicial hasta la posición final.

Como segunda etapa se insertaron los nuevos obstáculos al plano virtual, en tiempo de ejecución, para conseguir así una nueva ruta en caso interfiriera con la posición actual del agente y la ruta encontrada inicialmente.

A continuación, en las figuras 2 y 3 se muestran los recorridos, tanto el obtenido después de ejecutar el simulador como el replanteado al encontrar un obstáculo, respectivamente.

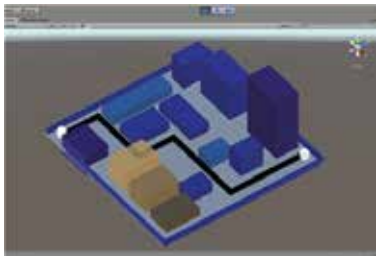

Figura 2. Simulación ejecutada y el camino más corto como resultado
Elaboración propia

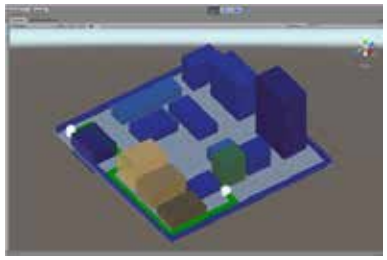

Figura 3. Replanteamiento del camino más corto debido a la presencia de un obstáculo colocado inicialmente
Elaboración propia

4.5 Presentar y evaluar los resultados generados por el algoritmo

En esta etapa se determinó el tiempo de ejecución experimental que necesitó el sistema para encontrar el camino más corto. Además, se validaron los resultados comparando el tiempo que tomó encontrar el camino al incrementar el tamaño del plano y la cantidad de obstáculos iniciales. Esto con el objetivo de comprobar el correcto funcionamiento y comportamiento del algoritmo.

Por otro lado, también se obtuvo como resultado el nuevo tiempo de ejecución al replantear la ruta debido a la presencia de nuevos obstáculos que interfirieron con el camino encontrado.

Cabe recalcar que la ejecución del simulador se realizó en una computadora con las siguientes características: procesador Intel Core i7-7630QM, CPU 2.60GHz, 8 GB de RAM, sistema operativo MacOS Mojave y tarjeta gráfica Radeon Pro 560X 4096 MB.

5. RESULTADOS

Posterior a ejecutar el simulador, en la prueba de concepto se logró calcular el tiempo de ejecución experimental requerido por el algoritmo para encontrar el camino más corto hacia la posición final.

Esto se realizó utilizando un paquete llamado System.Diagnostics el cual "proporciona clases que permiten interactuar con procesos del sistema, registros de eventos y contadores de rendimiento" (Microsoft, s. f.). De esta manera, se logró medir el tiempo transcurrido desde el inicio de la ejecución del simulador hasta la obtención del resultado.

Para la presentación de resultados estos se dividieron en tres criterios. El primer criterio mostró el camino más corto encontrado tomando en cuenta la posición del agente y la posición del punto de extracción seguro. Así, en la figura 4 se puede observar que el CPU requirió 22 milisegundos para encontrar la ruta, es decir, se demoró en calcular el camino más corto 0,022 segundos.

El segundo criterio presentó como resultado el tiempo de ejecución experimental requerido para replantear el nuevo camino más corto ante la existencia de obstáculos agregados manualmente que interfirieron con el camino encontrado inicialmente. En la figura 5 se puede observar que el CPU requirió 2 milisegundos para replantear la ruta más corta ante un obstáculo, es decir, se demoró 0,002 segundos en calcular el nuevo camino.

Por último, el tercer criterio mostró la presencia de obstáculos generados en tiempo real. Así, dio el resultado del tiempo de ejecución experimental requerido para encontrar el nuevo camino más corto ante la existencia de obstáculos generados dinámicamente. En la figura 6 se observa que, con 58 obstáculos dinámicos sobre el plano, el CPU requirió 3 milisegundos o 0,003 segundos para recalcular el nuevo camino más corto.

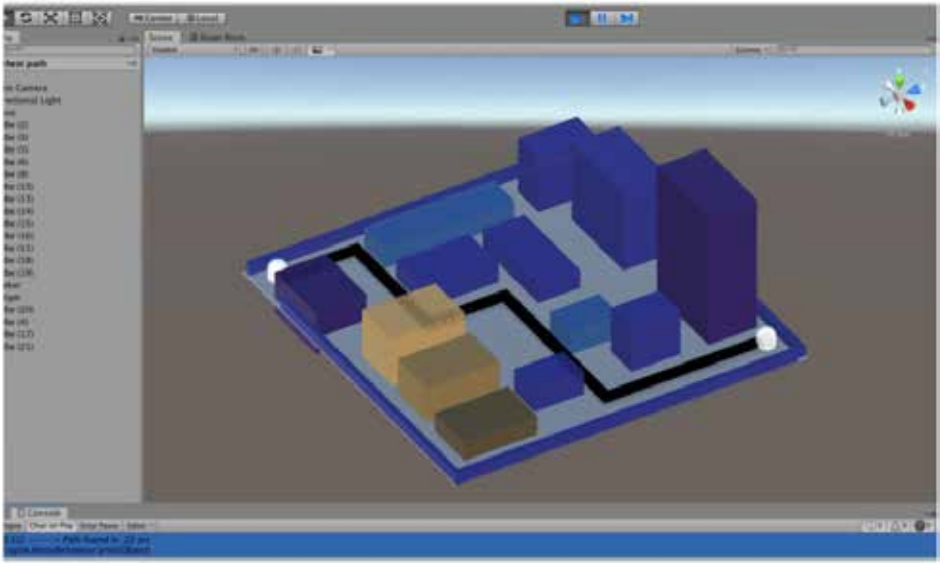

Figura 4. Tiempo requerido en milisegundos para encontrar el camino más corto inicial
Elaboración propia

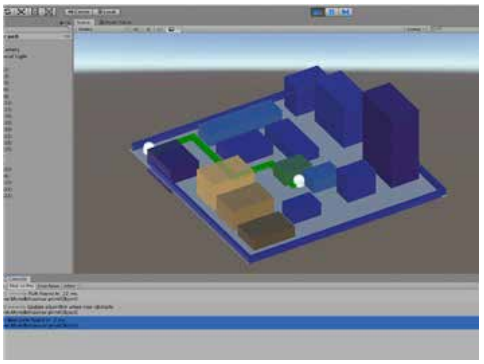

Figura 5. Estado del plano virtual con el nuevo camino más corto recalculado debido a la presencia de un obstáculo
Elaboración propia

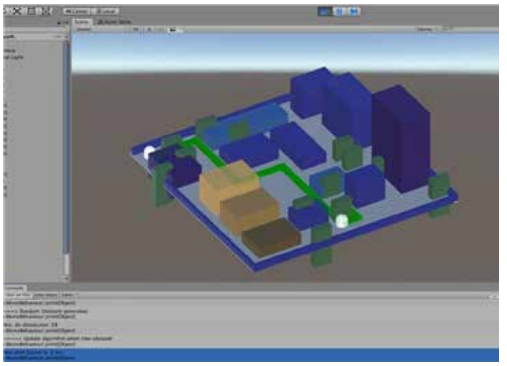

Figura 6. Estado del plano virtual con el nuevo camino más corto recalculado debido a la presencia de obstáculos dinámicos
Elaboración propia

A manera de resumen, se presenta una tabla comparativa con el objetivo de ilustrar de mejor manera los resultados obtenidos.

Tabla 1
Tabla comparativa de los resultados obtenidos con relación al tiempo de ejecución

Algoritmo	Tiempo de ejecución para encontrar el camino más corto inicialmente	Tiempo de ejecución para recalcular la ruta ante la presencia de un obstáculo agregado manualmente	Tiempo de ejecución para recalcular la ruta ante la presencia de obstáculos dinámicos
Algoritmo de búsqueda *D**	22 milisegundos (ms)	2 milisegundos (ms)	3 milisegundos (ms)

Elaboración propia

6. DISCUSIÓN DE LOS RESULTADOS

Para realizar una discusión y validación del funcionamiento del algoritmo y los resultados obtenidos en la prueba de concepto se tuvo en cuenta una técnica recurrentemente utilizada en la literatura revisada con relación a algoritmos más cortos. Esta técnica experimental consiste en realizar diferentes ejecuciones del algoritmo bajo diferentes escenarios.

Para la realización de la validación arbitrariamente se establecieron tres escenarios. El primero con una variación en el tamaño del plano, se realizaron ejecuciones con escalas de 5, 10 y 15 del plano y con 10 obstáculos iniciales.

El segundo escenario con una variación en la cantidad de obstáculos iniciales. Se realizaron ejecuciones con 10, 15 y 20 obstáculos, además, estas simulaciones fueron sobre un plano de escala 5.

Por último, se eligió una combinación de los seis escenarios propuestos. Este escenario ha sido aplicado en otras literaturas para ofrecer conclusiones acerca del algoritmo *D** ya que se guían del patrón aplicado en Stentz (1994). Así, se generaron cinco ejecuciones sobre un plano de escala 10 con 15 obstáculos iniciales. Con esto se comprobó la consistencia del tiempo de ejecución obtenido en las ejecuciones realizadas.

6.1 Primer escenario

A continuación, se presenta una tabla comparativa de los resultados obtenidos al ejecutar el simulador con los planos de tamaño de escala de 5, 10 y 15.

Tabla 2
Tabla comparativa de resultados para validar el comportamiento en diferentes dimensiones

Dimensiones del plano virtual	Tiempo de ejecución experimental obtenido (ms)
Escala 5	101 ms
Escala 10	543 ms
Escala 15	1474 ms

Elaboración propia

Como se puede observar en la tabla 2, el tiempo de ejecución experimental tiene un comportamiento incremental ya que este aumenta conforme se vayan aumentando las escalas del plano virtual.

6.2 Segundo escenario

A continuación, la tabla comparativa de los resultados obtenidos al ejecutar el simulador con 10, 15 y 20 obstáculos (véase la tabla 3).

Tabla 3
Tabla comparativa de resultados para validar el comportamiento con diferentes obstáculos

Cantidad de obstáculos sobre el plano virtual	Tiempo de ejecución experimental obtenido (ms)
10	99 ms
15	103 ms
20	104 ms

Elaboración propia

Como se puede observar en la tabla 3, el tiempo de ejecución experimental no varía de manera drástica comparado al escenario anterior. Así, se observa que la cantidad de obstáculos iniciales afectan al tiempo de ejecución en menor proporción comparado a la variación del tamaño del plano. Además, el tiempo de ejecución experimental depende de dónde se ubiquen los obstáculos iniciales en relación al punto de origen (agente) y el nodo objetivo.

6.3 Tercer escenario

Como último escenario, se muestra la tabla comparativa que abarca los resultados obtenidos al ejecutar un mismo escenario en cinco iteraciones (véase la tabla 4).

Tabla 4
Tabla comparativa de resultados para validar la consistencia del algoritmo al obtener el tiempo de ejecución experimental

Número de ejecución	Tiempo de ejecución experimental obtenido (ms)
1	612 ms
2	607 ms
3	621 ms
4	612 ms
5	605 ms

Elaboración propia

En base a lo observado en la tabla 4, al generar cinco ejecuciones del mismo escenario se obtienen tiempos de ejecución experimentales semejantes cuya variación no excede a 16 ms. De esta manera, se pudo inferir que el algoritmo *D** ofrece consistencia en la búsqueda del camino más corto. Por otro lado, estas variaciones del tiempo de ejecución pueden ser a causa del desempeño de la CPU en el instante en el que se ejecutó la simulación.

7. CONCLUSIONES

Como conclusión, se observa que en los diferentes escenarios realizados en el acápite 5, los resultados indicaron que al usar el algoritmo *D** se consiguió el camino más corto en solo 22 milisegundos, lo cual indica que en un entorno real de evacuación una persona conocería de manera precisa el camino más corto en menos de un segundo.

Asimismo, también se logró calcular el tiempo experimental que tomaría el algoritmo en recalcular el camino más corto y seguro en caso se evidencie la presencia de obstáculos que interfirieran en el recorrido. Así, al algoritmo solo le tomó 3 milisegundos en recalcular una segunda ruta eficiente. Esto indica que en un entorno real de evacuación una persona, que se encuentre recorriendo el camino sugerido inicialmente, podría conocer en menos de 0,001 segundos el camino más corto y seguro en caso de la presencia de obstáculos.

REFERENCIAS

Cornejo, M. B. (6 de junio del 2015). 60 % de instituciones sin plan para evacuar. *Correo.* Recuperado de https://diariocorreo.pe/peru/60-de-instituciones-sin-plan-para-evacuar-592886/

Haghani, M., y Sarvi, M. (2016). Human exit choice in crowded built environments: Investigating underlying behavioural differences between normal egress and emergency evacuations. *Fire Safety Journal*, *85*, 1-9. doi:10.1016/J.FIRESAF.2016.07.003

Hridi, A. P., Das, D., Anjum, M. M., y Das, T. (2016). Faster evacuation after disaster: Finding alternative routes using probable human behavior. *ACM DEV'16, Proceedings of the 7th Annual Symposium on Computing for Development*, 1-4. doi:10.1145/3001913.3006632

Iizuka, Y., y Iizuka, K. (2015). Disaster evacuation assistance system based on multi-agent cooperation. *HICSS'15: Proceedings of the 2015 48th Hawaii International Conference on System Sciences*, 173-181. doi:10.1109/HICSS.2015.30

Kocay, W., y Kreher, D. L. (s. f.). *Graphs, algorithms, and optimization*. Recuperado de https://scholar.google.com.pe/scholar?q=Kocay,+W.,+y+Kreher,+D.+L.+(s.+f.).+Graphs,+algorithms,+and+optimization.&hl=es&as_sdt=0&as_vis=1&oi=scholart

Koo, J., Kim, Y. S., Kim, B.-I., y Christensen, K. M. (2013). A comparative study of evacuation strategies for people with disabilities in high-rise building evacuation. *Expert Systems with Applications: An International Journal*, *40*(2), 408-417. doi:10.1016/j.eswa.2012.07.017

Masudur Rahman Al-Arif, S. M., Iftekharul Ferdous, A. H. M., y Hassan Nijami, S. (2012). Comparative study of different path planning algorithms: A water based rescue system. *International Journal of Computer Applications*, *39*(5), 975-8887. Recuperado de https://pdfs.semanticscholar.org/e59d/c7af0604fced7d124d4f755917725c7892e9.pdf

Microsoft. (s. f.). System. Diagnostics Namespace. Recuperado de https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics?view=netframework-4.8

Pelechano, N., y Badler, N. (2006). Modeling Crowd and Trained Leader Behavior during Building Evacuation. *IEEE Computer Graphics and Applications*, *26*(6), 80-86. doi:10.1109/MCG.2006.133

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'94)*, *4*, 3310-3317. Recuperado de https://www.ri.cmu.edu/publications/optimal-and-efficient-path-planning-for-partially-known-environments/

Thulasiraman, K., y Swamy, M. N. S. (2011). Basic concepts. En Thulasiraman y M. N. S. Swamy, *Graphs: Theory and Algorithms* (pp. 1-30). John Wiley & Sons. doi:10.1002/9781118033104.ch1

Ying, Z., Zi-min, Z., y Jian, C. (2017). EvacAgent: A Building Emergency Evacuation Simulation Model Based on Agent. *AIACT'17: Proceedings of the 2017 International Conference on Artificial Intelligence, Automation and Control Technologies*, 1-7. doi:10.1145/3080845.3080872