Ricardo Corbinaud Pérez

Universidad Tecnológica de Chile
ricardo.corbinaud@utem.cl

Nicolás Gárate González

Universidad Tecnológica de Chile
nicolasgarateg@gmail.com

RESOLUCIÓN Y ANÁLISIS DE PROBLEMAS DE BÚSQUEDA DE CAMINOS MÍNIMOS MEDIANTE ALGORITMOS METAHEURÍSTICOS

RESUMEN

Este paper aborda el problema de búsqueda de caminos mínimos entre dos puntos. Este tipo de problema es considerado como NP (No Polinomial), es decir no es resuelto efectivamente por algoritmos exactos. Y es en virtud de esto que se proponen alternativas entregadas por la inteligencia artificial para resolverlo. Particularmente en este documento se utiliza la metaheurística.

Mediante el método científico se utilizan dos técnicas metaheurísticas para resolver problemas de ruta mínima entre dos puntos en el plano XY con obstáculos: el "Algoritmo A*" y el "Algoritmo Colonización de Hormigas". El primero consta de una función heurística que es evaluada a cada instante y que entrega el siguiente paso a seguir de menor costo. El algoritmo "Colonización de Hormigas" está basado en el comportamiento de las hormigas que a través de las feromonas establecen un camino a seguir en busca del alimento.

El objetivo de este trabajo es realizar un estudio teórico y numérico, tendiente a evaluar el desempeño de las técnicas descritas anteriormente cuando son aplicadas a la resolución de problemas de búsqueda de caminos mínimos, y con base en los resultados concluir cuál de las técnicas expuestas es más recomendable utilizar para este tipo de problemas.

INTRODUCCIÓN

En el ámbito de Las Ciencias de la Computación existe una línea de investigación que abarca una serie de problemas complejos que hasta el día de hoy se encuentran sin tener una solución óptima, o dicho en términos computacionales, no poseen un algoritmo óptimo para resolver dicho problema. El hecho de poseer un algoritmo óptimo significa que no existe aún ningún otro algoritmo que para el problema en cuestión vaya a entregar una solución mejor a la encontrada por el primero.

Dentro de los problemas computacionales complejos mencionados anteriormente, están los que involucran optimización y/o combinatoria. Primero, porque la combinatoria en sí ya es un problema difícil de resolver (Roa Guzmán,

2000); y segundo, porque suelen acompañarse de factores específicos que restringen las soluciones. Es interesante notar que para un problema sin solución óptima existe una variedad de soluciones cercanas a lo óptimo que pueden ser calculadas por algoritmos ya existentes, como los metaheurísticos. Se dice que un algoritmo es metaheurístico cuando la solución no se determina en forma directa, sino mediante una estrategia maestra que guía y modifica otras heurísticas para producir soluciones más allá de las que se generan normalmente en una búsqueda de óptimos locales (Dorigo & Stützle, 2004).

El propósito de la investigación es realizar una implementación y comparación de dos algoritmos metaheurísticos: A* y ACO (Ant Colony Optimization). Ambos son aplicados a problemas de búsqueda de caminos mínimos y los resultados obtenidos son medidos cuantitativamente para ser analizados con el fin de determinar cuál presenta un mejor comportamiento en el problema planteado.

La primera parte del paper (materiales y métodos) consiste en establecer una base teórica de los algoritmos utilizados y los métodos involucrados durante el experimento, además de la especificación del problema puntual a resolver y los materiales necesarios. Con los materiales y métodos explicados en detalle, comienza la etapa de implementación y obtención de resultados para ser analizados en el punto "discusiones" del presente paper. Finalmente, con los resultados obtenidos viene todo el análisis cuantitativo sobre ellos y la comparación de ambas técnicas con el objetivo de establecer y concluir cuál de las dos entrega la mejor solución al problema.

MATERIALES Y MÉTODOS

Para el desarrollo de la investigación se utilizan dos métodos metaheurísticos, A* y ACO. Se entiende por metaheurística una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Las metaheurísticas proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los procedimientos estadísticos (Kelly & Osman, 1996).

Algoritmo A*

El algoritmo A* (conocido también como A estrella o asterisco) es un algoritmo de búsqueda que encuentra la ruta de menor costo entre dos puntos. Este algoritmo goza de una aceptable y continua implementación gracias a su desempeño y precisión. Fue descrito por primera vez en 1968 como una extensión del algoritmo de Dijkstra (1959) por Peter Hart, Nils Nilsson y Bertran Raphael, que expusieron que el A* lograba un mejor desempeño con respecto al tiempo usando heurísticas (Valenzuela, 2011).

Usa una función heurística para determinar el orden en que la búsqueda visita nodos en el árbol. La mencionada función es la suma de otras dos funciones: una función que indica el costo del camino seguido hasta un cierto nodo (denotada g(x)) y una estimación admisible de la distancia hasta la meta (h'(x)). La función de evaluación resulta entonces la ecuación (1):

$$f'(x) = g(x) + h'(x) \qquad (1)$$

Empezando en un nodo inicial dado, el algoritmo expande el nodo con el menor valor de f'(x). A* mantiene un conjunto de soluciones parciales almacenadas en una cola de prioridad. La prioridad asignada a un camino x viene determinada por la función f'(x). El proceso continúa hasta que una meta tiene un valor f'(x) menor que cualquier otro nodo en la cola (o hasta que el árbol ha sido completamente recorrido).

Suponga a alguien que quiere ir desde el punto "A" hasta el punto "B". Asuma también que un muro separa estos dos puntos tal como se muestra en la Ilustración 1. La clave para determinar qué cuadrados se usan para resolver el camino está en la ecuación (I), donde: g(x) = el costo de movimiento para ir desde el punto inicial "A" a un cierto cuadro del mapa, siguiendo el camino generado para llegar allí; h'(x) = el costo de movimiento estimado para ir desde ese cuadro del mapa hasta el punto "B", también conocida como la heurística. El camino se genera por ir repetidamente a través de una lista abierta y eligiendo el cuadrado con la puntuación f'(x) más baja. Suponga que se tiene un costo de 1 para los movimientos horizontales y verticales, y 1,4 para los diagonales. Ahora que se tiene calculado el costo g(x) mediante un camino específico hasta cierto cuadrado, la forma de resolver el costo g del cuadrado es coger el costo g(x) de su padre, y luego añadirle 1 o 1,4 dependiendo de si está en diagonal u ortogonal con respecto a ese cuadro padre. El método que se usa en este trabajo para estimar el valor de h'(x) es la distancia euclidiana desde un punto hacia el destino. Cabe señalar que cuando se calcula h'(x), se ignora cualquier obstáculo que intervenga en el camino y es una estimación de la distancia que queda, no de la distancia actual, es por eso que se llama heurística. Al sumar g(x) y h'(x) se obtiene f'(x). El resultado del primer paso en la búsqueda puede verse en la Ilustración 1. Las puntuaciones f'(x), g(x) y h'(x) están escritas en cada cuadrado. De los 9 cuadros iniciales, se dejan 8 en la lista abierta después de que el cuadrado inicial fuera incluido en la lista cerrada. Se elige el valor de f'(x) más bajo de todos aquellos que estén en la lista abierta. De estos, el que tiene el costo f(x) más bajo es el de la derecha del cuadro inicial, con un f(x) de 4. Así que se selecciona este cuadrado como el siguiente a seguir.

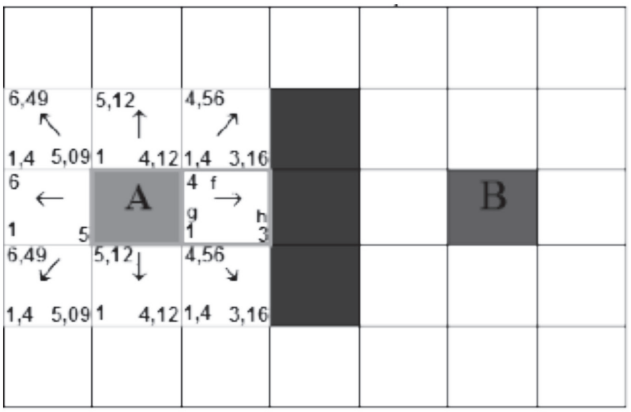


ILUSTRACIÓN 1 PRIMER MOVIMIENTO DE A*

Luego el cuadrado seleccionado se saca de la lista abierta y se añade a la lista cerrada. Se comprueban todos los cuadrados adyacentes, ignorando aquellos que estén en la lista cerrada o que sean intransitables, para añadir los cuadros a la lista abierta si no están ya en esa lista. Luego se hace al cuadro seleccionado el "padre" de los nuevos cuadros. Si el cuadro adyacente ya está en la lista abierta, se comprueba si el camino a ese cuadro es mejor que el actual. En otras palabras, se comprueba que la g(x) de ese cuadro es más baja que la del que se está usando para ir allí. Si no es así, no se hace nada. Por otro lado, si el costo g(x) del nuevo camino es más bajo, se cambia el padre del cuadro adyacente al cuadro seleccionado. Finalmente, se recalcula f' y la g de ese cuadrado. El proceso termina cuando se encuentra el cuadro objetivo, una vez hecho

esto se va desde éste hacia atrás siguiendo los padres hasta el inicio como se muestra en la Ilustración 2

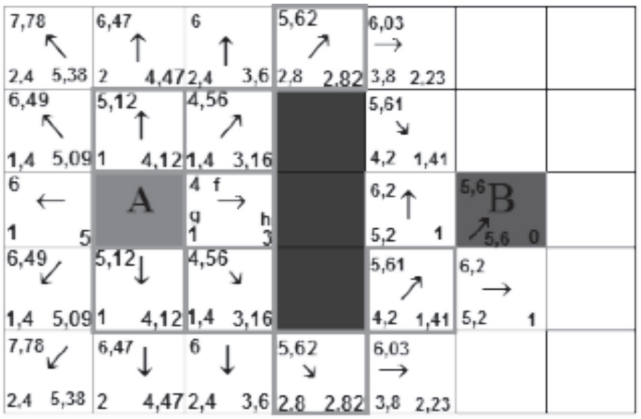


ILUSTRACIÓN 2 MOVIMIENTOS A*

Algoritmo ACO (Ant Colony Optimization)

Es un modelo inspirado en el comportamiento de colonias de hormigas reales. Estudios realizados explican cómo animales casi ciegos, como son las hormigas, son capaces de seguir la ruta más corta en su camino de ida y vuelta entre la colonia y una fuente de abastecimiento. Esto es debido a que las hormigas pueden "transmitirse información" entre ellas gracias a que cada una, al desplazarse, va dejando un rastro de una sustancia llamada feromona a lo largo del camino seguido. Así, mientras una hormiga aislada se mueve de forma esencialmente aleatoria, los "agentes" de una colonia de hormigas detectan el rastro de feromona dejado por otras hormigas y tienden a seguir dicho rastro. Éstas, a su vez, van dejando su propia feromona a lo largo del camino recorrido y, por tanto, lo hacen más atractivo, puesto que se ha reforzado el rastro de feromona. Sin embargo, la feromona también se va evaporando con el paso del tiempo provocando que el rastro de feromona sufra, por otro lado, cierto debilitamiento. En definitiva, puede decirse que el proceso se caracteriza por una retroalimentación positiva, en la que la probabilidad con la que una hormiga escoge un camino aumenta con el número de hormigas que previamente hayan elegido el mismo camino (Barcos, Rodríguez, Álvarez, & Robusté, 2002).

Las hormigas siguen el camino cuya probabilidad sea mayor, la misma que se expresa mediante la ecuación II (Michalewicz & Fogel, 2002):

$$P_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}}, & si\ j \in N_i^k \\ 0, & en\ otro\ caso \end{cases}$$

Donde P_{ij}^{k} es la probabilidad de que la hormiga k vaya del nodo i al nodo j;, η_{ij} es la información heurística, es decir, conocimiento previo;, N_i^k es el posible vecindario de una hormiga k cuando se encuentra en la ciudad i;, τ_{ij} es el rastro de feromona, ; α y β son parámetros que determinan la influencia de la feromona y la heurística, respectivamente.

En la actualización de las feromonas, el nivel de ésta en los caminos prometedores aumenta y disminuye en los caminos no tan buenos. Primero, se reducen todos los valores de feromona

por medio del proceso de evaporación. Luego, se incrementa el nivel de feromona al conjunto de soluciones buenas. Se utiliza la siguiente ecuación (Michalewicz & Fogel, 2002):

$$\tau_{ij}=(1-\rho)\ \tau_{ij}+\Delta\tau_{ij}$$

donde:

$\rho \in (0,1]$ Es el coeficiente que representa la evaporación del rastro de feromona.

$\Delta\tau_{ij}=\sum_{(k=1)}^{\text{NroHormigas}} \Delta^k \tau_{ij}$ Es la acumulación de rastro, proporcional a la calidad de las soluciones.

$$\Delta^k \tau_{ij} = \begin{cases} 1/L_k & , si\ la\ hormiga\ k\ utilizó\ la\ arista\ i,j\ en\ su\ recorrido \\ 0 & , \quad en\ otro\ caso \end{cases}$$

Representación

Una representación bidimensional (matricial) es el modelo computacional que se utiliza con dimensiones 10, 50, 100, 250 y 500 nodos de ancho y alto, con un nivel de obstáculos del 30% del total de nodos del mapa. Es decir, para el mapa de 10 x 10 se tienen 30 nodos obstáculos con el cuidado de que el borde superior y derecho del mapa queden despejados de manera que a lo menos exista un camino entre el origen y destino y evitar así mapas sin rutas exitosas. El origen de la ruta será el nodo ubicado en (0,0) y el destino el ubicado en el otro extremo del mapa, como se muestra en la Ilustración 3.

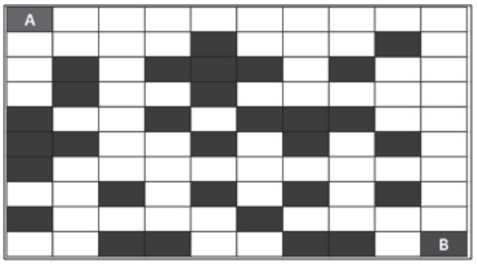


ILUSTRACIÓN 3 MAPA DE DIMENSIÓN 10 CON 30% DE OBSTÁCULOS

La localización de los obstáculos al interior de cada mapa se realiza de forma aleatoria, con la condición de que no sean el nodo inicio, fin ni los del borde superior y derecho como se explicó en el párrafo anterior. Además, los movimientos al interior del mapa tienen los siguientes valores:

Ortogonales: 1 unidad
Diagonales: 1,4142 unidades

La importancia de esto último radica en que la distancia recorrida de cada algoritmo se obtiene de la suma de los movimientos realizados al interior del mapa.

Pasando a las implementaciones de los algoritmos y el problema, se utiliza el lenguaje de programación C++ en Dev C++ para trabajar bajo la plataforma Windows 7 en un Notebook Sony VAIO con procesador Intel i3 de 2.3 GHz y 4 GB de memoria RAM.

RESULTADOS

En este punto se exhiben los resultados obtenidos para cada tamaño de mapa con cada algoritmo. Es importante mencionar que en el algoritmo ACO, al poseer este parámetros de regulación[1], se utilizan variaciones con el fin de encontrar una combinación que presente resultados aceptables. Por su parte, el algoritmo A* es una única versión. Se miden las siguientes variables: unidades de distancia recorrida, tiempo de ejecución y espacio de trabajo.

Algoritmo A*

La Tabla 1 muestra en su columna izquierda el tamaño (dimensión) de los distintos mapas utilizados, y en la columna derecha la distancia recorrida desde el origen al destino.

1 Cantidad de hormigas, nivel de evaporación de feromonas, etc.

Tamaño Mapa	Unidades Recorridas
10	15,071
50	72,811
100	150,55
250	374,4
500	747,87

TABLA 1 RESULTADOS DISTANCIA RECORRIDA A*

La Tabla 2 representa los resultados obtenidos del tiempo de ejecución de algoritmo para cada mapa, la columna izquierda entrega el tamaño del mapa utilizado y la derecha el tiempo de ejecución que tardó el algoritmo en encontrar el camino hacia el destino.

Tamaño Mapa	Tiempo ejecución (S)
10	0
50	0,01
100	0,237
250	2,196
500	17,803

TABLA 2 RESULTADOS TIEMPO DE EJECUCIÓN A*

Por último, la Tabla 3 presenta el espacio de trabajo utilizado durante la ejecución del algoritmo A* para cada mapa.

Tamaño Mapa	Espacio Trabajo (KB)
10	2068
50	2116
100	2352
250	3884
500	7876

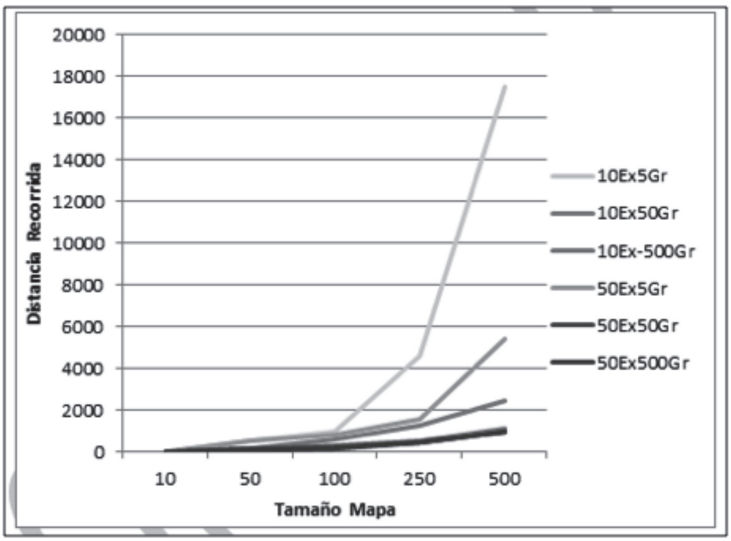
TABLA 3 RESULTADOS ESPACIO DE TRABAJO A*

Algoritmo ACO

Las primeras pruebas para este algoritmo se realizan con los siguientes valores:

Exploradoras: 10 y 50.
Repeticiones: 5, 50 y 500.
α: 1
β: 1
Evaporación feromona: 0,1

Y se obtienen los resultados expuestos en el gráfico de la Ilustración 4



ILUSTRACION 4 COMPARACION DISTANCIA RECORRIDA ACO 10 V/S 50 EXPLORADORAS

En vista de que la calidad de las soluciones (distancia recorrida) que entregan las 50 exploradoras es mejor que las de 10, se utilizan en un nuevo experimento 50 exploradoras pero con tasas de evaporación iguales a 10, 40 y 70%, con α=1, 5 y 10, 50 y 500 repeticiones. Así se obtienen los resultados del gráfico de la Ilustración 5:

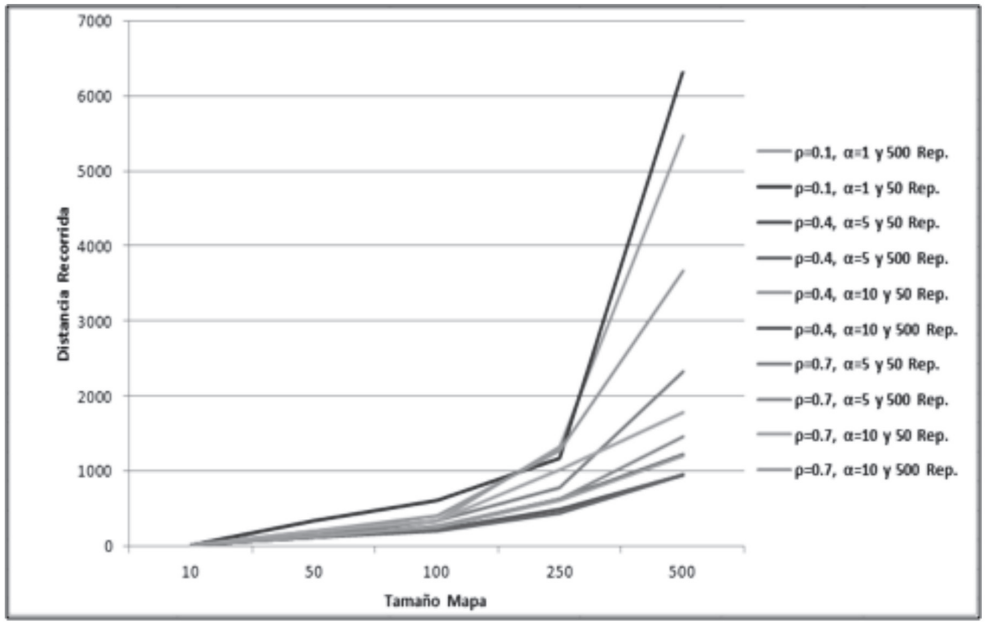


ILUSTRACIÓN 5 COMPARACIÓN VARIACIONES DE ALGORITMO ACO

Comparación A*-ACO

De todas las versiones del algoritmo ACO utilizadas, la que presenta mejores resultados es aquella que tiene tasa de evaporación del 40% con α=5 y 500 repeticiones. Las distancias recorridas de esta versión son comparadas con las del algoritmo A* y las ilustraciones 6, 7 y 8 representan los gráficos de comparación:

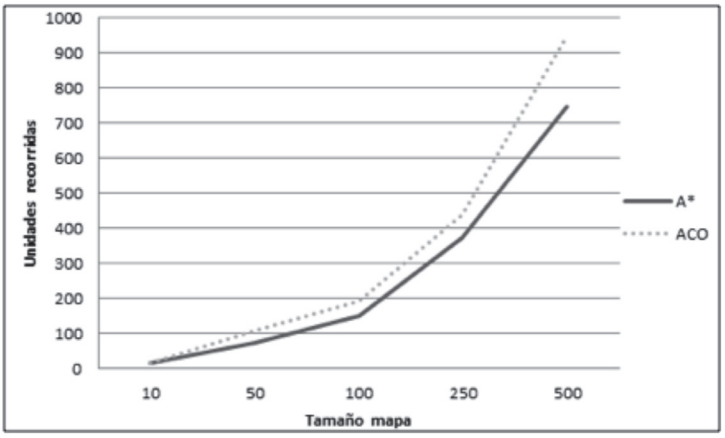


ILUSTRACIÓN 6 GRÁFICO FINAL DISTANCIA RECORRIDA ACO V/S A*

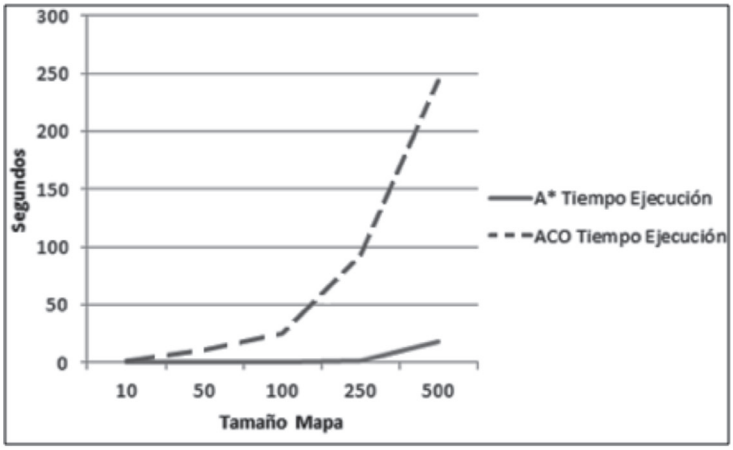


ILUSTRACIÓN 7 GRÁFICO FINAL TIEMPO DE EJECUCIÓN ACO V/S A*

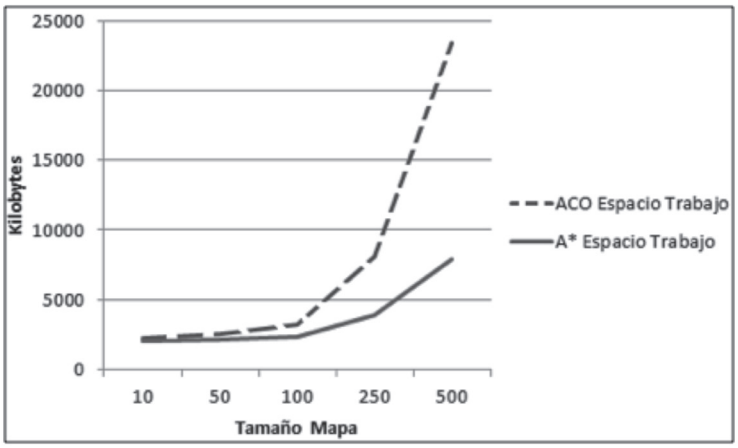


ILUSTRACIÓN 8 GRÁFICO FINAL ESPACIO DE TRABAJO ACO V/S A*

DISCUSIONES

Una interrogante que surge luego de ejecutar el algoritmo ACO con 10 y 50 exploradoras es si realmente existe alguna diferencia entre ellos o entregan resultados similares. Para clarificar esto, se utiliza la Ilustración 4 en la cual se muestra que las 50 exploradoras descubren caminos de menor distancia en comparación a las 10, tanto para las 5 y 50 como para las 500 repeticiones. Al realizar la comparación de todas las variaciones del ACO se obtiene un gráfico como el representado en la Ilustración 5, en el que la variación que presenta mejores resultados es la que tiene una tasa de evaporación del 40% con α=5 y 500 repeticiones. Las distancias recorridas de esta versión son comparadas con las del algoritmo A* y las Ilustraciones 6, 7 y 8 representan los gráficos de comparación en donde se clarifica la ventaja de A* sobre ACO con porcentaje de superioridad que va desde un 17,06% para el mapa de tamaño 250 hasta un 51,25% en el mapa de dimensión 50, obteniendo en promedio soluciones 29,14% más cortas que ACO. En cuanto a tiempo de ejecución, se muestra que el algoritmo A* es notablemente

más rápido que el ACO y, finalmente, la última comparación corresponde al espacio de trabajo utilizado por cada algoritmo, en donde la curva de ACO en la Ilustración 8 evidencia que ACO utiliza más memoria que A*.

El algoritmo A* de igual forma vence en los experimentos realizados al algoritmo ACO, en todas sus dimensiones, rapidez, calidad de la solución y espacio de trabajo.

CONCLUSIONES

La utilización de métodos de búsqueda meta-heurísticos, en particular ACO, no presenta mejores resultados en comparación al Algoritmo A*, pero no obstante esto último se aprecia una mejora en los resultados de ACO al calibrar los valores de los parámetros del algoritmo, presentando resultados cercanos al algoritmo A* cuando aumentan la importancia del nivel de feromonas y el coeficiente de evaporación. Esto significa que mientras más rápida sea la evaporación del rastro de feromona existente en el mapa y la importancia que se le asigna a este mismo, los resultados debiesen mejorar en comparación a tasas de evaporaciones bajas y lentas.

Ahora bien, se ha dicho que ACO presenta soluciones de baja calidad y lentas en relación a A*, pero existe una versión de ACO que es más rápida que A*. Aunque la calidad de las soluciones no es buena, se destaca dicha rapidez si el contexto del problema involucra encontrar soluciones en poco tiempo, independientemente de la calidad de la solución, como por ejemplo en situaciones críticas como la movilización de algún vehículo de emergencia, aunque habrá que evaluar si es mejor obtener una solución no suficientemente buena en un tiempo de espera bajo o esperar un poco más y obtener una solución mejor.

No cabe duda que A* es un algoritmo meta-heurístico bastante rápido, que consume pocos recursos y las soluciones obtenidas son de buena calidad, es por esto que se recomienda utilizar A* en problemas de búsqueda caminos mínimos en perjuicio de utilizar ACO.

REFERENCIAS

Barcos, L., Rodríguez, V., Álvarez, M. J., & Robusté, F. (2002). Algoritmo basado en la optimización mediante colonias dehormigas para la resolución del problema del transporte de carga desde varios orígenes a varios destinos. Santander, España.

Dorigo, M., & Stützle, T. (2004). Ant Colony Optimization. Massachusetts, Estado Unidos.

Kelly, J., & Osman, I. (1996). Meta-Heuristics: Theory & Applications. Massachusetts, Estados Unidos: Kluwer Academic Publishers.

Michalewicz, Z., & Fogel, D. B. (2002). How to Solve It: Modern Heuristics. Berlín, Alemania: Springer.

Roa Guzmán, R. (2000). Razonamiento combinatorio en estudiantes con preparación matemática avanzada. Granada, España.

Valenzuela, R. (2011). Algoritmo A-star (asterisco). Nogales, México.