Hormigas Urbanas

Nicolás Gabriel Giuliano y Joaquín González Budiño
Facultad de Ingeniería y Ciencias Hídricas – nsgiuliano@gmail.com, joa_gzb@hotmail.com

***Resumen*— En la actualidad no existen muchos servicios que permiten visualizar mapas y planear rutas considerando el uso de transportes urbanos en una ciudad. Una posible solución a esta necesidad es mediante el uso de algoritmos de búsqueda tradicionales, como ser costo uniforme, cuyo costo es muy eficiente para mapas pequeños. La desventaja surge cuando el tamaño del mapa aumenta porque los costos en tiempo y memoria crecen exponencialmente.**

Se propone aquí el uso de métodos de Inteligencia Colectiva inspirados en el comportamiento de hormigas (que poseen un costo espacial acotado -cantidad de hormigas-) para resolver este tipo de problemas, evitar las desventajas recién mencionadas y obtener resultados igual de razonables.

El objetivo será comparar los distintos métodos de implementación de colonia de hormigas, seleccionar la alternativa metódica mas adecuada y estudiar su comportamiento en base a variaciones en sus parámetros característicos. El resultado de esta comparación será mostrar qué algoritmo (derivado del ACO) es más recomendado para diferentes problemas en particular.

IV. Introducción

El problema que se quiere resolver consiste en encontrar el camino más corto a partir de dos puntos (inicial y final) de un mapa urbano, teniendo en cuenta los recorridos de los colectivos y sus respectivas direcciones.
Este problema es un derivado del Problema del Agente Viajero (Travelling Salesman Problem -TSP- , donde se requiere recorrer todos los puntos de un mapa con menor costo) o la búsqueda de camino optimo multiobjetivo (en donde a partir de un mapa se quiere encontrar el camino que pase por una serie de puntos de interés).

Al ser técnicas metaheurísticas, no se garantiza la mejor solución posible, pero sí una muy buena en un rango de soluciones que va encontrando el algoritmo.

Se establece una serie de requisitos que deberá cumplir el problema:

- A través de la representación de un mapa urbano, se formará un grafo cuyos nodos serán las esquinas y las aristas serán las calles que conectan pares de esquinas (nodos).
- Las aristas tendrán un peso, que será modelado como el tiempo en minutos.
- Las aristas pueden ser unidireccionales (recorridos de colectivos) o bidireccionales (desplazamiento a pie), y para aristas similares su peso puede ser diferente.
- Se puede elegir cualquier nodo de la red como origen o destino.
- Existirán dos grafos claves, el primero para el movimiento en un mapa a pie, y el segundo para el movimiento en un mapa en colectivos. Ambos grafos son independientes entre si.
- Se ha de contemplar que los colectivos se detienen cada dos cuadras mayormente. Subir al colectivo también implica un coste en tiempo de espera y un coste en pagar por el servicio de utilizarlo. Este ultimo se traduce a precio por tiempo.

V. Configuración del problema

Se consideran ciertos aspectos que son datos claves del problema:

- El mapa es un dato y, por lo tanto, se conoce desde antes de realizar cálculos, cual es el origen y destino que tiene un usuario a la hora de pedir una ruta al sistema.
- Se ha preguntado a trece personas a partir de cuantas cuadras considera tomar el colectivo para viajar. En promedio el resultado fué que a partir de ocho cuadras o mas, las personas optan por usar un medio de transporte publico para moverse por una ciudad. También se ha preguntado cual es su preferencia a la hora de tomar un viaje (el coste de un boleto o el tiempo de viaje). Casi todos han respondido que la cuestión de tiempo es mas importante a la hora de viajar. Es por esto que este trabajo se centra en brindar una respuesta optima en sentido temporal, pues el coste de un boleto pasa a ser una variable secundaria.

El problema se modela en tres etapas:

1. La primera etapa consiste en determinar si es factible ir hacia una parada, esperar el colectivo y pagar por su servicio. O bien realizar el viaje unicamente a pie, pues es mas rápido. Esto está muy ligado a la distancia que existe del origen al punto destino. Entonces se hace una evaluación del coste aproximado entre solo caminar u optar por el colectivo. Se considera que se esperan 3 minutos (como caso ideal) para la llegada del próximo colectivo, 1 minuto para caminar una cuadra y el costo de boleto de un colectivo, que lo traducimos a tiempo (2 minutos). Así, se tiene un coste de subida a cualquier colectivo igual a 5 minutos. Con esta lógica, se cumple el promedio estudiando anteriormente: a partir de ocho cuadras el sistema opta por evaluar que linea de colectivo es la mas adecuada para ir al destino deseado. Caso contrario (menos de 8 cuadras), el ruteo se realiza con posibilidad unicamente de caminar.
2. La segunda etapa consiste es hacer un estudio para determinar cual es la parada mas cercana a la que debe concurrir el usuario para subirse a un colectivo que lo acercarse a su destino. También determinar en que parada bajarse. Se estudia estas paradas para cada linea de colectivo, su recorrido y la distancia de sus paradas respecto al origen(donde se ubica el usuario) y destino (hacia donde se dirige). En este punto, el mejor caso se da cuando la parada de subida y la de bajada corresponden a la misma linea de colectivos. El peor de los casos se da cuando ambas paradas corresponden a distintas lineas.

Se seleccionan solo las lineas de colectivos y sus combinaciones, en base a la parada de subida y bajada del usuario en su trayecto en colectivo. Con esto se forma un segundo grafo que involucra unicamente los recorridos de los colectivos seleccionados, donde los lugares en donde se debe hacer un transbordo y contemplan un peso temporal igual a 5 (peso asociado a subir un colectivo, a pesar del transbordo).

3. La tercera y ultima etapa simplemente aplica los algoritmos de búsqueda sobre ambos grafos, para trazar el recorrido completo: del nodo origen→parada de subida, parada de subida→ parada de bajada, parada de bajada→ nodo destino. En el primer y ultimo caso se utilizan los grafos bidireccionales que conectan cada esquina entre si, para trazar el camino a pie. En el segundo caso se utiliza el grafo construido especialmente para trazar el camino en colectivos (y sus posibles combinaciones).

III. Modelos de Hormigas Bio-Inspirados

La idea en común para todos los algoritmos probados, es la siguiente: Primero, el comportamiento de las hormigas es aleatorio, es decir, eligen (o no) recorrer una camino en base al nivel de feromona que exista en cada alternativa. Al encontrar una fuente de comida (en el problema esto se traduce a que llego a destino), vuelven liberando una cantidad de feromonas proporcional a la cantidad de comida (en nuestro caso, como queremos minimizar el tiempo, la cantidad de comida es inversamente proporcional al tiempo). Luego, las demás hormigas se ven influenciadas por el rastro de feromona reforzado por la hormiga anterior. Hay una probabilidad mayor de que sigan éste camino. Este rastro se va reforzando cuantas más hormigas pasen por él, y además se simula una evaporación del rastro como sucede en la naturaleza.

El depósito de feromonas se puede realizar de dos formas:

- Una vez que cada hormiga del hormiguero ha construido su solución, se procede a actualizar los rastros de feromonas en cada camino encontrado en función de la calidad de su solución. Así el proceso pasa a llamarse actualización fuera de línea de rastros de feromona (o actualización global de feromona).
- Se puede depositar una cantidad de feromona a medida que las hormigas están construyendo su solución (actualización en línea).

Además, se puede ponderar el rastro de feromonas de distintas maneras. En el caso extremo, si se pondera a la cantidad de feromonas con un exponente cero, aquellos nodos con una preferencia heuristica mejor tienen una mayor probabilidad de ser escogidos, haciendo al algoritmo muy similar a un algoritmo voraz probabilístico clásico (con múltiples puntos de partida en caso de que las hormigas estén situadas en nodos distintos al comienzo de cada iteración).

Sin embargo, si se pondera al tiempo con un exponente cero, solo se tienen en cuenta los rastros de feromona para guiar el proceso constructivo, lo que puede causar un rápido estancamiento, esto es, una situación en la que los rastros de feromona asociados a una solución son ligeramente superiores al resto, provocando por tanto que las hormigas siempre construyan las mismas soluciones, normalmente mínimos locales.

IV. Descripción de métodos

A. Algoritmo de Colonia de Hormigas

Es el algoritmo clásico modificado al detalle de que si la hormiga se pierde (no encuentra un camino valido en el mapa), se establece su ruta con un tiempo infinito. El tiempo será un parámetro para la función de fitness.

Todas aquellas hormigas que se consideren perdidas (tiempo de resolución infinito) no depositaran feromona en la iteración actual.

Por el contrario, cada hormiga que encuentre un camino valido, deposita feromona:

$$r_{rs} \leftarrow r_{rs} + p * r_{rs}$$

Donde p es la cantidad de feromona que se deposita, función de fitness y r_{rs} la arista que conecta los nodos r y s que forman parte del camino solución S encontrada por la hormiga. Luego se realiza una evaporación constante a nivel global de la feromona:

$$r_{rs} \leftarrow (1 - \rho)\, r_{rs}$$

siendo $\rho \in [0,1]$ el factor de evaporación. Una hormiga *k* puede ir al siguiente nodo con una probabilidad:

$$p_{rs}^{k} = \begin{cases} \dfrac{[r_{rs}]^{\alpha} x [\eta_{rs}]^{\beta}}{\sum_{u \in N_r^k} [r_{rs}]^{\alpha} x [\eta_{rs}]^{\beta}} & si\ s \in N_k^r \\ 0, & en\ otro\ caso \end{cases}$$

α y β son parámetros que ponderan la importancia de la heurística utilizada y los valores de feromona detectados.

r_{rs} representa el rastro de feromona entre los nodos r y s.
η_{rs} representa el valor del tiempo (en min) de una arista.

B. Sistema de Colonia de Hormigas (ACS)

Este algoritmo cada hormiga realiza una actualización local en los rastros de feromona.

$$\tau_{rs} \leftarrow (1 - \rho)\, x\, \tau_{rs} + \rho * \tau_0$$

Donde $\phi \in (0,1]$ es un segundo parámetro de decremento de feromona. También se suma el detalle de que sólo considera una hormiga concreta: la que generó la mejor solución global. Esta hormiga será la que nuevamente depositará feromona, haciendo que su camino sea destacable respecto a los otros. En otras palabras, la mejor hormiga deposita dos veces feromona.

Este algoritmo suma otra funcionalidad más: una regla proporcional pseudo-aleatoria. Esta cambia la manera de avanzar entre los nodos y la asignación de probabilidades de explotar los nodos. Sea k una hormiga situada en el nodo *r*, q_0 entre (0-1] y *q* un valor aleatorio en *[0,1]*, el siguiente nodo *s* se elige aleatoriamente mediante la siguiente distribución de probabilidad:

$$\text{Si } q \le q_0 \qquad p_{rs}^{k} = \begin{cases} 1, si \quad s = \arg\max_{u \in N_k(r)} \left\{ \tau_{ru} \times \eta_{ru}^{\beta} \right\} \\ 0, en \quad otro \quad caso \end{cases}$$

$$\text{Si } q > q_0 \qquad p_{rs}^{k} = \begin{cases} \dfrac{[r_{rs}]^{\alpha} \times [\eta_{rs}]^{\beta}}{\sum_{u \in N_r^k} [r_{ru}]^{\alpha} \times [\eta_{ru}]^{\beta}}, & si \quad s \in N_k(r) \\ 0, & en \quad otro \quad caso \end{cases}$$

c. *Sistema Mejor-Peor Hormiga*

Es una extensión del ACH con la diferencia que se refuerza la feromona del camino de la mejor hormiga y se castiga a la peor hormiga disminuyendo feromona de su camino de la siguiente manera:

$$\tau_{rs} \leftarrow \tau_{rs} + \rho \times f(C(S_{mejor-global})), \quad \forall a_{rs} \in S_{mejor-global}$$

$$\tau_{rs} \leftarrow (1-\rho) \times \tau_{rs} \quad \forall a_{rs} \in S_{peor-actual} \quad y \quad a_{rs} \notin S_{mejor-global}$$

Introduce, también, conceptos de computación evolutiva al realizar una mutación en las feromonas para evitar estancamiento en una solución (global o local) y aumentar las chances de encontrar la mejor solución. El rastro de feromona en todas las conexiones se muta con un operador y una probabilidad (a) de aplicar dicho operador. El rango de mutación *mut*(*it*, τumbral), que depende de la media(threshold) de los rastros de feromona en las transiciones de la mejor solución global:

$$\tau'_{rs} \leftarrow \begin{cases} \tau_{rs} + mut(it, \tau_{umbral}), & si \quad a = 0 \\ \tau_{rs} - mut(it, \tau_{umbral}), & si \quad a = 1 \end{cases}$$

$$mut(it, \tau_{threshold}) = \frac{it - it_r}{Nit - it_r} \cdot \sigma \cdot \tau_{threshold}$$

siendo Nit el máximo numero de iteraciones, σ es un valor al azar e it_r la ultima iteración donde se han reiniciado los rastros.

Se considera la reinicialización de los rastros de feromona cuando se estanca la búsqueda, lo que se lleva a cabo fijando cada rastro de feromona a τ_0.

C. Sistema de Hormigas Max-Min

Exhibe una alternativa al Sistema de Hormigas (AS), que difiere en tres aspectos clave:

- Es estilista en la deposición de feromona: solo deposita feromona la/s hormiga/s que haya encontrado la mejor solución en la iteración, o aquella que encontró la mejor solución desde el inicio de la ejecución
- acotar el nivel de feromona en el grafo entre un mínimo y un máximo para evitar el aumento desmesurado del valor de feromona en rutas no óptimas.
- Se inicializan los rastros de feromona al valor máximo, logrando que se haga una exploración mayor al inicio del algoritmo. Luego se realiza una evaporación global en todas las conexiones. El valor de feromona máximo se calcula en cada iteración:

$$\tau_{ij}^{max}(t) = \frac{1}{1-\rho} \frac{1}{f(s^{opt})}.$$

Mientras que el valor mínimo de feromona es constante. Cabe destacar por otro lado, que al dar la posibilidad de recorrer cualquier camino, es muy probable que al encontrar el camino deseado (u optimo) pero aun así, no todas las hormigas lo recorrerán, al tener acotado el valor de la feromona.

V. Implementación

Para la resolución del problema se requiere de un mapa. En este caso se decidió usar OpenStreetMap que provee la estructura del mapa en formato XML que indica cada esquina del mapa, sus conexiones, sentidos de calle,latitud, longitud.

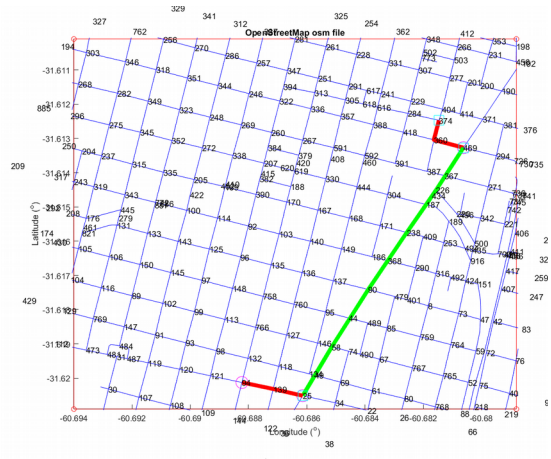
Para procesar esos archivos XML (de extensión .osm) y convertirlos en un grafo. Para ello, se utilizó una librería para MATLAB llamada OpenStreetMap functions, la cual permite cargar el mapa XML en un struct de MATLAB, cargar una matriz esparcida de adyacencia y poder graficar las calles.

Además de esto cargaremos un grafo similar a la matriz de adyacencia generada por la librería, e implementamos los recorridos de los colectivos.

De esta forma, ya se tiene todo lo necesario para aplicar los algoritmos. La ruta que se tiene como resultado de los algoritmos incluye las esquinas que sigue la ruta hallada.

A la librería actual se le agregó la representación de los sentidos de las calles mediante flechas (*mediante la función arrow obtenida de fuentes de internet*).

A continuación se mostrarán ejemplos de rutas graficadas mediante la librería:



Se observa la imagen la ejecución del algoritmo de sistema de hormigas. Las lineas rojas indican el tramo a pie, y la linea verde el tramo en colectivo. Nos brinca una idea de en que parada se debe esperar el colectivo, y en que parada hay que bajarse.

VI. Experimentos y Resultados

A continuación, se muestran los tiempos y resultados de los algoritmos mencionados anteriormente:

MAPA DOBLE DE TAMAÑO	ACH			MMAS		
	costo (mins)	tiempo resol(seg)	iteraciones	costo (mins)	tiempo resol(seg)	iteraciones
promedios	8,600	0,710	17,000	7,600	10,452	736,100
desv	1,370	0,138		0,316	1,737	

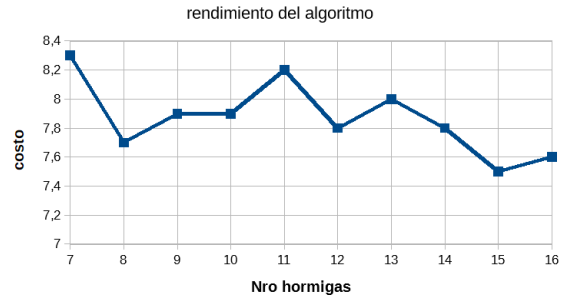
	ACS			BWAS		
	costo (mins)	tiempo resol(seg)	iteraciones	costo (mins)	tiempo resol(seg)	iteraciones
promedios	8,700	1,198	63,100	7,800	0,605	12,800
desv	1,398	0,304		0,483	0,060	

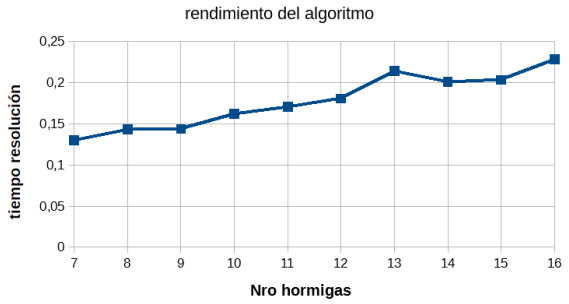
MAPA NORMAL	ACH			MMAS		
	costo (mins)	tiempo resol(seg)	iteraciones	costo (mins)	tiempo resol(seg)	iteraciones
promedios	8,100	0,116	18,200	7,500	11,871	523,200
desv	0,699	0,026		0,000	4,384	

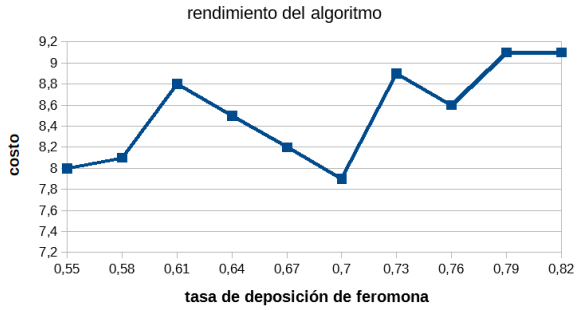
	ACS			BWAS		
	costo (mins)	tiempo resol(seg)	iteraciones	costo (mins)	tiempo resol(seg)	iteraciones
promedios	8,000	0,162	25,800	8,300	0,092	11,800
desv	0,972	0,062		0,632	0,019	

entre el mapa en el que se trabajo (1,2 km²) y un mapa con el doble de tamaño (3km²) el costo total de la solución hallada y el tiempo de resolución del algoritmo aumenta 2,5 veces. Mirando los resultados, podemos ver que MMAS si bien encuentra resultados óptimos casi siempre, el tiempo de solución es elevado. Debido a esto descartamos el uso de dicho método porque se considera una espera excesiva que el

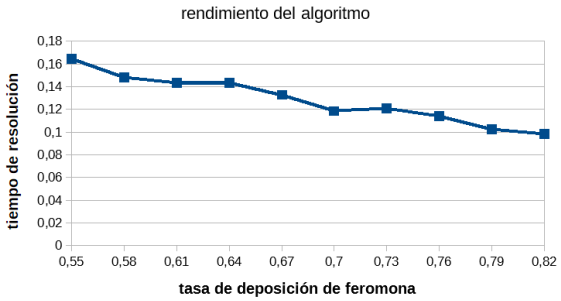
usuario hace para tener una respuesta favorable. Respecto a los demás algoritmos, se considera que BWAS resuelve el problema muy rápido, con resultados bastante favorables (siempre esta entre los 3 mejores). Por este motivo, lo seleccionamos como método favorito.
A continuación, se desea optimizar dicho algoritmo de Mejor Peor hormiga para aumentar las chances de obtener casi siempre el resultado optimo. Para esto se ha realizado un estudio de los parámetros del algoritmo (cantidad de hormigas y tasa de deposición de feromona). Se ha realizado 10 corridas del mismo algoritmo, con el mismo punto de origen y destino, para determinar el tiempo de resolución y costo de la solución promedio. Luego esto se fue repitiendo para distintos parámetros del algoritmos:







En base a esto, se concluye que para un total de 15 hormigas con una tasa de deposición de feromona de 70% el fitness total de la solución (y tasa de evaporación de 100%-70%=30% de feromona global y castigo), las soluciones serán optimas. Se ha realizado una nueva evaluación cambiando el destino y se concluyo que:



- el costo optimo era de 7 minutos.
- El tiempo de resolución de este costo optimo era de 0,112 segundos con 12 iteraciones en total.

Los resultados obtuvidos fueron:

BWAS resultados generales		
costo	tiempo	iteraciones
7,03	0,14	10,87
0,18	0,03	1,17

VII. Conclusiones Y Trabajos Futuros

En este trabajo se introdujeron varias técnicas para la obtención de una ruta en un mapa. Se sumo mucha importancia al tiempo en llegar hasta un destino y al tiempo de resolución del algoritmo. Para ello se ha estudiado los parámetros que minimizan el tiempo de resolución del algoritmo, y maximizan la obtención de rutas optimas.

Con estos resultados, el tamaño del mapa será una cuestión menor, solo implicaría a partir de aquí aplicar optimizaciones sobre el grafo.

Como trabajo futuro se puede plantear la incorporación de otros medios de transporte, como pueden ser autos, bicicletas, etc.

Otra posible idea sería evaluar este mismo trabajo pero en mapas mucho más grandes, que puedan abarcar varias ciudades, y ver cómo mejora la performance de los algoritmos de hormigas frente al de costo uniforme. En el presente trabajo esto no se pudo realizar ya que implicaba procesar matrices más grandes y de mayor costo computacional.

Referencias

[1] Jesús Rodríguez García. Tesis. *Análisis de algoritmos basados en colonia de hormigas en problemas de camino mínimo.* Universidad Carlos III de Madrid, 2010

[2] Thomas Stützle, Holger H. Hoos. *MAX–MIN Ant System. Future Generation Computer Systems.* 2000

[3] Oscar Cordón, Iñaki Fernández de Viana, Francisco Herrera. *Analisys of the Best-Worst Ant System and its Variants on the TSP. Mathware & Soft Computing*. University of Granada. España 2002. IRIDIA, Université Libre de Bruxelles & Computer Science Department, University of British Columbia.

[4] Xianmin Wei. *Improvement and Implementation of Best-worst Ant Colony Algorithm.* Research Journal of Applied Sciences, Engineering and Technology. School of Computer Engineering, Weifang University, Weifang 261061, China. 2013.